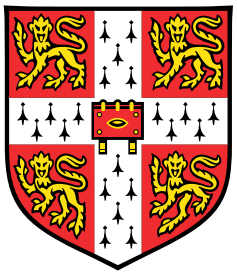
Improving Representation Learning through Generative Modeling

From Diffusion Models to Riemannian Geometry



Georgios Batzolis

Department of Applied Mathematics and Theoretical Physics
University of Cambridge

This dissertation is submitted for the degree of
Doctor of Philosophy

Churchill College December 2024

I would like to dedicate this thesis to my loving parents ...

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Georgios Batzolis
December 2024

Acknowledgements

And I would like to acknowledge ...

Abstract

This is where you write your abstract ...

Notation

This section introduces the notations used throughout the thesis, combining statistical concepts with elements from differential and Riemannian geometry. For details on Riemannian geometry, see [21, 27, 116, 169].

Statistical Notation

Random variables are denoted by capital letters (X, Y), with their associated probabilities written as $P(X), P(Y)$, and their densities as p_X, p_Y. Calligraphic letters $(\mathcal{X}, \mathcal{Y})$ represent sample spaces, while specific samples are denoted by lowercase bold letters $(\mathbf{x}, \mathbf{y})$. Given two random variables X and Y, the conditional probability of Y given X is denoted by $P(Y \mid X)$, and its density by $p_{Y|X}$. Unless otherwise specified, all probabilities are assumed to admit a density.

Simplified Statistical Notation. In certain chapters, as noted at their outset, we adopt a streamlined statistical notation to improve clarity and simplify the presentation of formulas. For instance, probability distributions are denoted using only the argument of their density. For a random variable X_t with a realization x_t, we write:

$$p(x_t) := p_{X_t}(x_t).$$

Riemannian Geometry Notation

Let $\mathcal{M}$ denote a smooth manifold. The space of smooth functions on $\mathcal{M}$ is denoted by $C^\infty(\mathcal{M})$. The *tangent space* at $\mathbf{p} \in \mathcal{M}$, defined as the space of all *derivations* at $\mathbf{p}$, is written as $\mathcal{T}_\mathbf{p}\mathcal{M}$. Elements of $\mathcal{T}_\mathbf{p}\mathcal{M}$, called *tangent vectors*, are denoted by $\Xi_\mathbf{p} \in \mathcal{T}_\mathbf{p}\mathcal{M}$. The *tangent bundle* of $\mathcal{M}$ is $\mathcal{T}\mathcal{M}$, and smooth vector fields—smooth sections of the tangent bundle—are written as $\mathcal{X}(\mathcal{M}) \subset \mathcal{T}\mathcal{M}$.

A smooth manifold $\mathcal{M}$ becomes a *Riemannian manifold* if equipped with a smoothly varying *metric tensor field*, $(\cdot,\cdot) : \mathcal{X}(\mathcal{M}) \times \mathcal{X}(\mathcal{M}) \to C^\infty(\mathcal{M})$. This induces a *(Riemannian) metric*, $d_\mathcal{M} : \mathcal{M} \times \mathcal{M} \to \mathbb{R}$. The metric tensor also defines a unique affine connection, the *Levi-Civita connection*, denoted by $\nabla_{(\cdot)}(\cdot) : \mathcal{X}(\mathcal{M}) \times \mathcal{X}(\mathcal{M}) \to \mathcal{X}(\mathcal{M})$.

The Levi-Civita connection allows the definition of *geodesics*. A geodesic between two points $\mathbf{p}, \mathbf{q} \in \mathcal{M}$ is a curve $\gamma_{\mathbf{p},\mathbf{q}} : [0,1] \to \mathcal{M}$ of minimal length connecting $\mathbf{p}$ to $\mathbf{q}$. For a tangent vector $\Xi_\mathbf{p} \in \mathcal{T}_\mathbf{p}\mathcal{M}$, the geodesic with initial velocity $\dot{\gamma}_{\mathbf{p},\Xi_\mathbf{p}}(0) = \Xi_\mathbf{p}$ defines the

exponential map, $\exp_{\mathbf{p}} : \mathscr{D}_{\mathbf{p}} \to \mathscr{M}$, as:

$$\exp_{\mathbf{p}}(\Xi_{\mathbf{p}}) := \gamma_{\mathbf{p},\Xi_{\mathbf{p}}}(1),$$

where $\mathscr{D}_{\mathbf{p}} \subset \mathscr{T}_{\mathbf{p}}\mathscr{M}$ is the domain on which $\gamma_{\mathbf{p},\Xi_{\mathbf{p}}}(1)$ is well-defined. The *logarithmic map*, $\log_{\mathbf{p}} : \exp(\mathscr{D}'_{\mathbf{p}}) \to \mathscr{D}'_{\mathbf{p}}$, is the inverse of $\exp_{\mathbf{p}}$, valid on the domain where $\exp_{\mathbf{p}}$ is a diffeomorphism.

Finally, let $(\mathscr{M}, (\cdot,\cdot))$ be a d-dimensional Riemannian manifold, and $\mathscr{N}$ a d-dimensional smooth manifold. If $\phi : \mathscr{N} \to \mathscr{M}$ is a diffeomorphism, the *pullback metric* on $\mathscr{N}$ is defined as:

$$(\Xi, \Phi)^{\phi} := (D_{(\cdot)}\phi[\Xi_{(\cdot)}], D_{(\cdot)}\phi[\Phi_{(\cdot)}])_{\phi(\cdot)}, \tag{1}$$

where $D_{\mathbf{p}}\phi : \mathscr{T}_{\mathbf{p}}\mathscr{N} \to \mathscr{T}_{\phi(\mathbf{p})}\mathscr{M}$ is the differential of ϕ. This pullback metric equips $\mathscr{N}$ with a Riemannian structure, $(\mathscr{N}, (\cdot,\cdot)^{\phi})$, preserving geometric properties from $(\mathscr{M}, (\cdot,\cdot))$. In this thesis, pullback mappings follow 1, denoted with ϕ as a superscript, e.g., $d^{\phi}_{\mathscr{N}}(\mathbf{p},\mathbf{q})$, $\gamma^{\phi}_{\mathbf{p},\mathbf{q}}$, $\exp^{\phi}_{\mathbf{p}}(\Xi_{\mathbf{p}})$, and $\log^{\phi}_{\mathbf{p}}\mathbf{q}$.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

Over the past decade, artificial intelligence (AI) has undergone a transformative evolution, driven by groundbreaking advancements in generative modeling and self-supervised learning (SSL). These two fields have redefined our ability to model complex data distributions and leverage vast amounts of unlabeled data, leading to significant breakthroughs in areas such as media synthesis [94, 159, 51, 105], multimodal learning [157, 215, 207], and scientific discovery [91, 191, 163, 222].

Generative modeling focuses on learning the underlying probability distributions of data to create realistic and diverse samples. Innovations such as Generative Adversarial Networks (GANs) [62], diffusion models [77, 185], and normalizing flows [161, 146] have set new benchmarks in quality and applicability, enabling realistic image generation [94], lifelike video creation [51], natural-sounding audio synthesis [105], and text-to-image generation [159]. Beyond creative media, these models play a critical role in scientific applications, such as protein structure prediction [91, 8], AI-assisted drug discovery [191, 57], and materials science [24], where they accelerate research by generating novel molecules and exploring chemical spaces. Furthermore, they address challenges in inverse problems, including super-resolution [168, 209] and inpainting [195, 126].

Self-supervised learning, on the other hand, has emerged as a powerful approach to harness the vast reservoirs of unlabeled data by uncovering its inherent structure. SSL enables models to learn meaningful representations that are crucial for downstream tasks, often serving as the backbone of state-of-the-art systems across diverse domains. These representations can be extracted using generative methods, which model the data distribution, or discriminative methods, such as contrastive learning [33], which focus on distinguishing between similar and dissimilar data samples.

Self-supervised learning has had a profound impact on foundational technologies. In natural language processing, it powers groundbreaking models like BERT [40] and GPT

[23], which underpin conversational AI and advanced language understanding systems. In computer vision, SSL reduces dependence on labeled data while delivering robust, scalable visual representations, driving improvements in tasks like object recognition and image segmentation. These advancements underscore SSL's versatility and its pivotal role in advancing AI across domains and modalities. Additionally, SSL has been transformative in multimodal learning, enabling seamless alignment and integration across modalities such as text, images, and audio. This capability has unlocked advanced tasks like generating descriptive image captions [81], synchronizing audio with video [3], retrieving relevant content across formats [192], and enhancing representation learning for multimodal datasets [157].

In this thesis, we focus on the theoretical and computational aspects of generative modeling and generative self-supervised learning. To provide the necessary background, the remainder of this introduction is divided into two sections: an overview of generative modeling (1.1) and an overview of self-supervised learning (1.2).

1.1 Generative Modeling

1.1.1 The Problem of Generative Modeling

Generative modeling sits at the forefront of machine learning, enabling us to capture the underlying structure of complex data and generate new, realistic samples. Whether it's creating lifelike images, composing music, or designing novel molecules, generative models aim to understand and replicate the distribution of data they're trained on.

Formally, let's consider a dataset $\mathscr{D} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \subset \mathbb{R}^d$, comprising N independent samples drawn from an unknown data-generating distribution $P(X)$, which we refer to as the *target distribution*. The central goal of generative modeling is to create a model that approximates $P(X)$ as closely as possible based solely on these samples.

To learn the distribution $P(X)$, we seek a computationally tractable approximation $\hat{P}(X)$ based on the dataset $\mathscr{D}$ and our assumptions about $P(X)$. We measure how close $\hat{P}(X)$ is to $P(X)$ using a divergence $D(\hat{P}(X), P(X))$, such as the Kullback-Leibler divergence or the Wasserstein distance. Our goal is to minimize this divergence:

$$P^*(X) = \arg\min_{\hat{P}(X)} D(\hat{P}(X), P(X)).$$

Typically, we parameterize $\hat{P}(X)$ with a finite-dimensional vector $\theta \in \mathbb{R}^p$, forming a family of models $P_\theta(X)$ and transforming the problem into optimizing over parameters:

$$\theta^* = \arg\min_{\theta} D(P_\theta(X), P(X)).$$

For practicality, $P_\theta(X)$ must be computationally tractable; we should be able to efficiently sample from it or evaluate its probability density function. Different generative modeling approaches balance tractability and fidelity to $P(X)$, leading to various methods suitable for different applications.

By tackling this problem, we aim not only to reproduce the data we observe but also to uncover the underlying mechanisms that generate it. This deep understanding empowers us to create models that can generate new, unseen data that is indistinguishable from real-world samples, thereby expanding the horizons of what machines can learn and create.

1.1.2 Why Deep Learning for Generative Modeling?

Non-parametric methods such as Kernel Density Estimation (KDE) have been widely used for modeling probability distributions in low-dimensional settings due to their flexibility in estimating the underlying density without assuming a specific parametric form [176]. However, scaling KDE to high-dimensional data is problematic. The sample complexity of KDE scales exponentially with the number of feature dimensions d, making it impractical for high-dimensional tasks [211]. Specifically, to achieve a target estimation error ε, the number of samples n must satisfy:

$$n \geq \mathcal{O}\left(\varepsilon^{-\frac{4+d}{4}}\right).$$

[175] This exponential dependence on d implies that even for moderate dimensions, the required sample size becomes infeasibly large, rendering KDE and similar methods ineffective for generative modeling in high dimensions.

In many real-world applications, however, data does not uniformly occupy the high-dimensional ambient space but is concentrated near a lower-dimensional manifold—a concept known as the *manifold hypothesis* [54]. For example, natural images, although high-dimensional when represented by pixel values, often lie on manifolds of much lower intrinsic dimensionality due to dependencies and correlations among pixels [165]. This implies that the effective complexity of the data is determined by the manifold's dimension k, where $k \ll d$.

While the manifold hypothesis suggests a reduction in effective dimensionality, leveraging this property requires models that can explicitly capture the underlying manifold structure. Non-parametric methods like KDE do not inherently account for the manifold and treat each dimension independently, failing to exploit the dependencies among features [198].

As a result, they still suffer from high sample complexity despite the data's lower intrinsic dimensionality.

Parametric models with appropriate inductive biases can overcome this limitation by incorporating assumptions about the data's structure into the learning process. Deep learning architectures are particularly well-suited for this task due to their ability to integrate domain-specific inductive biases through their design [111]. For instance, Convolutional Neural Networks (CNNs) employ local connectivity and weight sharing to capture spatial hierarchies and local patterns in image data [112]. Similarly, Recurrent Neural Networks (RNNs) are designed to model temporal dependencies in sequential data [79].

By aligning model architecture with the intrinsic properties of the data manifold, deep learning models effectively reduce the hypothesis space to functions consistent with the data's structure, enhancing learning efficiency and reducing the required number of samples [153]. This structural approach explains why deep learning–based methods have become indispensable for generative modeling in high dimensions [60, 102].

1.1.3 Deep Learning for Generative Modeling

In the past decade, deep learning has introduced flexible architectures that can effectively learn complex data distributions. This chapter reviews five main types of generative models: Generative Adversarial Networks (GANs), Normalizing Flows, Auto-Regressive Models, Energy-Based Models, and Diffusion Models, each with unique strengths and limitations.

Generative Adversarial Networks (GANs)

Introduced by Goodfellow et al. [62], Generative Adversarial Networks (GANs) consist of two neural networks: a *generator* G and a *discriminator* D, trained simultaneously in a minimax game. The generator maps samples $\mathbf{z}$ from a simple prior distribution $P(Z)$ (e.g., a standard normal distribution) to the data space $\mathbb{R}^d$, producing synthetic data $G(\mathbf{z})$. The discriminator outputs a probability $D(X) \in [0,1]$ indicating whether a sample X is real (from the training data) or generated (from G).

The training objective is formulated as a two-player minimax game with the value function $V(D,G)$:

$$\min_G \max_D V(D,G) = \mathbb{E}_{X\sim P(X)}[\log D(X)] + \mathbb{E}_{Z\sim P(Z)}[\log(1-D(G(Z)))]$$

where $P(X)$ is the target distribution represented by the training set $\mathscr{D}$.

Training Process GANs are trained through an adversarial process where the generator and discriminator are updated iteratively to improve against each other. The discriminator D learns to better distinguish real data from generated data, while the generator G aims to produce data that can fool D. This training continues until the generator produces samples indistinguishable from real data ($P_G(X) = P(X)$).

There are various formulations of the value function designed to enhance training stability and performance. For instance, the Wasserstein GAN (WGAN) [5] modifies the original objective using the Earth Mover's distance to address issues like mode collapse and improve convergence.

Strengths GANs have several notable strengths. One of their primary advantages is their ability to produce **high-quality samples** that are visually realistic and sharp, especially in image generation tasks. Additionally, GANs offer **efficient sampling**; once the generator is trained, it can rapidly produce new samples through a single forward pass, without the need for iterative sampling procedures common in other generative models.

Weaknesses Despite their strengths, GANs also present significant weaknesses. A major challenge is the **training instability** inherent in their adversarial setup, which can lead to issues such as mode collapse—where the generator produces limited varieties of samples—vanishing gradients, and oscillatory behavior during training. Moreover, GANs **do not provide an explicit density** function or likelihood estimate for the generated data, limiting their applicability in tasks that require probabilistic interpretations or quantitative assessments of sample likelihood.

Normalizing Flows

Normalizing flows, introduced by Rezende and Mohamed [161], are a versatile class of generative models that provide a unified framework for both data generation and explicit likelihood estimation. These models transform a simple base distribution into a complex target distribution using a sequence of invertible and differentiable mappings, enabling exact computation of likelihoods. Normalizing flows are trained by directly maximizing the log-likelihood of the observed data, leveraging the change of variables formula to compute densities in closed form. This combination of flexibility, tractability, and theoretical soundness makes normalizing flows a powerful tool for modeling complex probability distributions.

Given a latent variable $Z \sim P(Z)$ in latent space $\mathscr{Z}$ and an invertible function $G^\theta : \mathscr{Z} \to \mathscr{X}$ parameterized by θ, the model defines the random variable $X = G^\theta(Z)$ in data space $\mathscr{X}$

to approximate the data distribution $P(X)$. The transformation G^θ maps samples from the base distribution to samples resembling the observed data.

Change of Variables To compute the probability density function (pdf) of X, the change of variables formula is employed. For an invertible and differentiable transformation G^θ, with inverse $F^\theta = (G^\theta)^{-1}$, the pdf of X is given by:

$$p_X^\theta(\mathbf{x}) = p_Z\left(F^\theta(\mathbf{x})\right)\left|\det\left(\frac{\partial F^\theta(\mathbf{x})}{\partial \mathbf{x}}\right)\right|,$$

where $\frac{\partial F^\theta(\mathbf{x})}{\partial \mathbf{x}}$ is the Jacobian matrix of F^θ at point $\mathbf{x}$, and det denotes the determinant. Taking the logarithm yields:

$$\log p_X^\theta(\mathbf{x}) = \log p_Z\left(F^\theta(\mathbf{x})\right) + \log\left|\det\left(\frac{\partial F^\theta(\mathbf{x})}{\partial \mathbf{x}}\right)\right|.$$

This formulation enables exact computation of the log-likelihood, which is crucial for effective training.

Composing Transformations In practice, the transformation G^θ is constructed by composing a sequence of K invertible and differentiable transformations:

$$G^\theta = G_K \circ G_{K-1} \circ \cdots \circ G_1,$$

where each G_k is an invertible function, and the composition is defined as $G^\theta = G_K(G_{K-1}(\ldots G_1(z)))$. The inverses of these transformations are denoted as $F_k = G_k^{-1}$.

For a data point $\mathbf{x}$, intermediate variables are defined by recursively applying the inverses:

$$\mathbf{h}_0 = \mathbf{x}, \quad \mathbf{h}_k = F_k(\mathbf{h}_{k-1}), \quad \text{for } k = 1, \ldots, K.$$

The log-density of $\mathbf{x}$ can then be expressed as:

$$\log p_X^\theta(\mathbf{x}) = \log p_Z(\mathbf{h}_K) + \sum_{k=1}^{K} \log\left|\det\left(\frac{\partial F_k(\mathbf{h}_{k-1})}{\partial \mathbf{h}_{k-1}}\right)\right|.$$

Efficient Computation A crucial aspect of designing normalizing flows is ensuring that the composing functions G_k are, typically, analytically invertible to allow for fast sampling. Additionally, the calculation of the determinant of the Jacobian must be tractable to enable efficient training via the change of variables formula. This is often achieved by designing the transformations so that the resulting Jacobian matrices have a lower triangular form or

another structure that simplifies the computation of the determinant. By constructing G_k in this manner, normalizing flows facilitate efficient evaluation of both the forward and inverse transformations, as well as the log-determinant of the Jacobian, thereby enabling effective training and fast sampling from the model.

Training Objective The parameters θ of the normalizing flow are optimized by maximizing the likelihood of the observed data $\{\mathbf{x}_i\}_{i=1}^N$, which is equivalent to minimizing the negative log-likelihood:

$$\theta^* = \arg\min_\theta \sum_{i=1}^{N} -\log p_X^\theta(\mathbf{x}_i).$$

By leveraging the exact computation of the log-likelihood, normalizing flows can be trained using standard gradient-based optimization methods, ensuring stable and efficient convergence.

Strengths Normalizing flows offer several advantages. A primary strength is their ability to perform **exact density computation**, allowing for tractable and exact likelihood evaluations. This facilitates stable training via maximum likelihood estimation [146]. Furthermore, they provide **efficient sampling**; once trained, new samples can be generated rapidly by transforming latent samples through G^θ. The analytical invertibility of G^θ and efficient computation of Jacobian determinants make both training and sampling computationally efficient.

Weaknesses Despite their strengths, normalizing flows have limitations. The requirement for invertibility and tractable Jacobians imposes constraints on the choice of transformations, potentially leading to **limited expressivity** [146]. This can hinder the model's ability to capture complex data distributions, especially those with intricate structures. Additionally, normalizing flows can encounter **topological limitations and numerical stability issues** when modeling targets with complicated topologies. In such cases, the model may become numerically non-invertible to approximate the target distribution closely, leading to stability problems and reducing the effectiveness of the flow [15, 37].

Normalizing flows, while powerful, **lack a semantically meaningful lower-dimensional latent space**, unlike GANs and Variational Auto-Encoders (VAEs). GANs and VAEs are designed with a lower-dimensional latent space, explicitly chosen a priori, enabling their effective use in downstream tasks such as compression, image-text alignment, and editing. In contrast, the latent space of normalizing flows is inherently tied to the ambient space's

dimensionality, limiting its direct interpretability. In Chapter 5, we address this limitation by adapting normalizing flows to acquire an interpretable latent space. Our approach combines an anisotropic base distribution with isometry regularization, drawing on principles from pullback Riemannian geometry. This training paradigm constructs a latent space where semantic significance is encoded by the variances learned in the base distribution. Latent dimensions capturing meaningful semantic information are characterized by high variances, while those encoding noise exhibit very low variance, enabling a clear distinction between essential and extraneous information.

Auto-Regressive Models

Auto-regressive models are a class of generative models that factorize the joint probability density of high-dimensional data into a product of conditional densities, leveraging the chain rule of probability. Notable examples include PixelCNN and PixelRNN [142], WaveNet [141], and more recently, Transformer-based models [203], which have been applied across various domains such as natural language processing, image generation, and audio synthesis.

Model Formulation Given a data vector $\mathbf{x} = [x_1, x_2, \ldots, x_d]$ in $\mathbb{R}^d$, the joint density $p_X(\mathbf{x})$ is decomposed into a product of conditional densities:

$$p_X(\mathbf{x}) = \prod_{i=1}^{d} p_{X_i|X_{1:i-1}}(x_i \mid x_{1:i-1}),$$

where $X_{1:i-1}$ denotes all preceding random variables in a predefined ordering, and $x_{1:i-1}$ represents their corresponding realizations. This factorization enables the model to capture complex dependencies between variables by modeling each variable conditioned on all previous ones.

Modeling Conditional Densities Each conditional density $p_{X_i|X_{1:i-1}}(x_i \mid x_{1:i-1})$ is modeled using a neural network that takes $x_{1:i-1}$ as input and outputs the parameters of the density for x_i. For instance, in PixelCNN, convolutional neural networks with masked filters are employed to ensure that the prediction of a pixel x_i depends only on pixels $x_{1:i-1}$ that come before it in the chosen ordering.

The masking technique is critical; it zeroes out connections in the network to prevent information from "future" variables influencing the prediction of the current variable, thus maintaining the auto-regressive property.

Training Objective The parameters θ of the auto-regressive model are optimized by maximizing the likelihood of the observed data $\{\mathbf{x}_n\}_{n=1}^N$, which is equivalent to minimizing the negative log-likelihood:

$$\theta^* = \arg\min_\theta \sum_{n=1}^{N} -\log p_{X,\theta}(\mathbf{x}_n) = \arg\min_\theta \sum_{n=1}^{N} \sum_{i=1}^{d} -\log p_{X_i|X_{1:i-1},\theta}(x_{n,i} \mid x_{n,1:i-1}),$$

where $x_{n,i}$ represents the i-th variable in the n-th observed data sample, and $x_{n,1:i-1}$ denotes its preceding realizations.

Sampling Process Sampling from auto-regressive models is inherently sequential. To generate a new data sample $\mathbf{x}$, the model starts by sampling x_1 from $p_{X_1}(x_1)$. Then, for $i = 2$ to d, each x_i is sampled from $p_{X_i|X_{1:i-1}}(x_i \mid x_{1:i-1})$, using the previously sampled values. This process continues until all variables have been generated.

Strengths Auto-regressive models have several notable strengths. A primary advantage is their ability to perform **exact likelihood computation**, providing an exact calculation of the data likelihood without any approximation. This facilitates stable training via maximum likelihood estimation, ensuring that the model accurately captures the underlying data density. Additionally, by conditioning each variable on all previous ones, auto-regressive models are highly effective at **modeling complex dependencies** in sequential or structured data. This makes them particularly well-suited for tasks such as language modeling, image generation, and audio synthesis, where capturing intricate patterns and dependencies is crucial. Furthermore, auto-regressive models offer **flexibility in handling different data types**; they can model both discrete and continuous data by appropriately choosing the form of the conditional densities, such as using categorical densities for discrete variables or Gaussian densities for continuous ones.

Weaknesses Despite these strengths, auto-regressive models also present significant weaknesses. A major limitation is the **slow sampling process** inherent in their sequential nature. Generating a single data sample requires sampling each variable one at a time, conditioned on the previously generated variables. This makes the sampling process computationally intensive and time-consuming, especially for high-dimensional data like high-resolution images or long sequences, where the number of variables d is large. Moreover, the sequential dependency of variables **limits parallelization during sampling**, as each variable depends on the outcome of the previous ones, preventing the use of parallel computation techniques

to speed up sample generation. This can be particularly problematic when rapid generation of many samples is required.

Energy-Based Models (EBMs)

Energy-Based Models (EBMs) [113] define an unnormalized probability density over data using an energy function $E_\theta(\mathbf{x})$. The density of a data point $\mathbf{x} \in \mathscr{X}$ is given as:

$$p_\theta(\mathbf{x}) = \frac{\exp(-E_\theta(\mathbf{x}))}{Z_\theta},$$

where $Z_\theta = \int_{\mathscr{X}} \exp(-E_\theta(\mathbf{x}))\, d\mathbf{x}$ is the partition function, a normalizing constant that ensures $p_\theta(\mathbf{x})$ integrates to 1 over $\mathscr{X}$. However, computing Z_θ is typically intractable for high-dimensional data, posing challenges for direct optimization and sampling.

Training Objective EBMs are trained to assign low energy values to samples drawn from the data distribution (positive samples) and high energy values to samples from the model distribution (negative samples). A commonly used training objective is based on contrastive divergence [75], which minimizes the expected energy of positive samples while maximizing the expected energy of negative samples:

$$\mathscr{L}(\theta) = \mathbb{E}_{\mathbf{x}^+ \sim P(X)}[E_\theta(\mathbf{x}^+)] - \mathbb{E}_{\mathbf{x}^- \sim P_\theta(X)}[E_\theta(\mathbf{x}^-)],$$

where $\mathbf{x}^+$ are realizations from the data distribution $P(X)$, and $\mathbf{x}^-$ are realizations from the model distribution $P_\theta(X)$.

In order to obtain negative samples $\mathbf{x}^-$, a standard approach is to use *Markov Chain Monte Carlo (MCMC)* methods, with Langevin Dynamics [108, 134] being a widely adopted technique. Langevin Dynamics is a stochastic gradient-based method that iteratively updates a sample $\mathbf{x}_t$ to move toward regions of higher probability density under the unnormalized model density $p_\theta(\mathbf{x})$. The iterative update rule is given by:

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta \nabla_{\mathbf{x}} E_\theta(\mathbf{x}_t) + \sqrt{2\eta}\varepsilon_t,$$

where $\eta > 0$ denotes the step size, and $\varepsilon_t \sim \mathscr{N}(\mathbf{0}, I)$ represents Gaussian noise added to incorporate stochasticity into the updates. The term $-\nabla_{\mathbf{x}} E_\theta(\mathbf{x}_t)$ drives the sample $\mathbf{x}_t$ toward regions of lower energy, effectively encouraging exploration of the high-density regions of $p_\theta(\mathbf{x})$, while the stochastic noise ε_t prevents the chain from getting trapped in local minima.

This approach ensures that the generated negative samples $\mathbf{x}^-$ align with the model's current energy landscape. The addition of noise also maintains ergodicity, which is crucial for

exploring the sample space comprehensively. In the limit of small step sizes η and sufficient iterations, the chain converges to the target distribution $P_\theta(X)$.

Sampling Process Sampling from EBMs follows the same procedure as obtaining negative samples. Since direct sampling is infeasible due to the intractability of Z_θ, MCMC methods like Langevin Dynamics are used to approximate samples from the target distribution $P_\theta(X)$. This iterative process converges to high-density regions of $P_\theta(X)$, making it suitable for generation tasks. However, the need for multiple iterations results in a slow sampling process, particularly for high-dimensional data.

Strengths EBMs offer several notable strengths. One of their primary advantages is their **flexibility** in representing complex distributions without requiring explicit normalization, as they define an unnormalized probability distribution directly using an energy function [113]. This flexibility allows EBMs to focus on modeling the energy landscape of the data, making them highly adaptable to diverse data types and problem settings. Unlike other generative models, such as GANs or normalizing flows, which often require specific assumptions about the form of the probability distribution or rely on explicit normalization, EBMs can represent distributions of arbitrary complexity by appropriately designing the energy function.

Furthermore, EBMs exhibit strong **compositionality**, allowing seamless integration with other models and the incorporation of domain-specific knowledge such as constraints or priors [49, 66]. This property enables EBMs to handle applications requiring hybrid generative processes or strict adherence to domain rules. By encoding physical laws or chemical properties directly into the energy function, EBMs can generate outcomes that comply with these constraints, making them particularly effective in scientific modeling tasks like physical simulations and molecular design [49]. In structured prediction tasks, EBMs can impose priors to ensure outputs follow specific patterns or relationships, such as spatial coherence in semantic segmentation or hierarchical structures in knowledge graph synthesis [66, 113]. This compositional and flexible nature allows EBMs to excel in applications requiring constrained generative modeling or the integration of domain knowledge, where conventional generative models may struggle.

Weaknesses Despite their strengths, EBMs also face significant weaknesses. A major challenge is their **training difficulty**, primarily due to the reliance on MCMC methods for generating negative samples. MCMC sampling is computationally intensive, can suffer from slow convergence, and often requires careful tuning to ensure effective exploration of the energy landscape. Additionally, EBMs exhibit **slow sampling speed**, as generating a single

sample requires running iterative Langevin Dynamics or similar methods, which involve multiple gradient evaluations. This can hinder the scalability of EBMs for tasks requiring large-scale generation.

Diffusion and Score-Based Models

Diffusion models [77] and score-based generative models (SGMs) [186] are powerful generative frameworks that treat data generation as the process of reversing noise corruption. These models consist of two complementary components: a forward process that gradually perturbs data into a noise-like prior distribution and a reverse process that reconstructs the data by inverting this transformation. The forward process maps a complex data distribution into a simpler, analytically tractable distribution, typically the standard normal distribution $\mathcal{N}(0,\mathbf{I})$, while the reverse process transforms noise back into samples from the original data distribution.

In discrete-time models, such as Denoising Diffusion Probabilistic Models (DDPMs), this transformation occurs over a finite sequence of time steps, governed by a Markov chain. In contrast, continuous-time frameworks, such as Score-Based Generative Models (SGMs), model this progression using stochastic differential equations (SDEs). As we will see, these two frameworks are mathematically equivalent in the limit of infinite time steps. The reverse process, which lies at the heart of both frameworks, is the critical mechanism for generating high-quality samples.

Forward Process The forward process in diffusion and score-based generative models progressively corrupts the original data into noise, providing the foundation for learning the reverse process that generates data from noise. The design of the forward process ensures that over time, the data distribution transitions smoothly into a simple, analytically tractable distribution π, such as $\mathcal{N}(0,\mathbf{I})$.

- **DDPM (Discrete Time):**

  In the Denoising Diffusion Probabilistic Model (DDPM), the forward process is a Markov chain that adds Gaussian noise to the data at discrete time steps $t = 1,2,\ldots,N$. The transition kernel is given by:

  $$q_{X_t|X_{t-1}}(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}\left(\mathbf{x}_t; \sqrt{\alpha_t}\,\mathbf{x}_{t-1}, \beta_t\mathbf{I}\right),$$

  where:

  - $\alpha_t = 1-\beta_t$, the scaling factor applied to $\mathbf{x}_{t-1}$,

– $\beta_t \in (0,1)$, the predefined variance schedule controlling the amount of noise at step t,

This formulation allows the process to be expressed in terms of the initial data $\mathbf{x}_0$:

$$q_{X_t|X_0}(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}\right),$$

where $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$ is the cumulative product of α_s. This closed-form expression simplifies training and allows direct sampling from any step t.

- **SGM (Continuous Time):** The forward process in Score-Based Generative Models (SGM) is described by a continuous-time Stochastic Differential Equation (SDE) that progressively perturbs the data distribution with infinitesimal noise:

$$dX_t = f(X_t,t)\,dt + g(t)\,dW_t,$$

where:

– W_t is a standard Wiener process (Brownian motion),

– $f(X_t,t)$ is the drift coefficient, defining the deterministic evolution of the data distribution,

– $g(t)$ is the diffusion coefficient, controlling the scale of the stochastic noise.

The drift and diffusion coefficients $f(X_t,t)$ and $g(t)$ are carefully chosen so that, after a sufficiently large diffusion time T, the data distribution evolves into a distribution that is very close to a simple, analytically tractable distribution, typically the standard normal distribution $\mathcal{N}(\mathbf{0},\mathbf{I})$.

Popular choices for the forward SDE include:

– **Variance Preserving (VP) SDE:**

$$f(X_t,t) = -\frac{1}{2}\beta(t)X_t, \quad g(t) = \sqrt{\beta(t)},$$

where $\beta(t)$ is a predefined noise schedule that controls the rate of diffusion. This choice preserves the variance of the data distribution throughout the process. The limiting distribution as $t \to \infty$ is the standard normal distribution. In practice, after sufficient diffusion time T, the perturbed distribution $p_T \approx \mathcal{N}(\mathbf{0},\mathbf{I})$.

– **Variance Exploding (VE) SDE:**

$$f(X_t,t) = \mathbf{0}, \quad g(t) = \sqrt{\frac{d[\sigma^2(t)]}{dt}},$$

where $\sigma(t)$ is a monotonically increasing function that scales the noise over time. In this formulation, the variance of the data distribution "explodes." After sufficient diffusion time T, the perturbed distribution $p_T \approx \mathcal{N}(\mathbf{0}, \sigma(T)^2\mathbf{I})$.

These formulations ensure a smooth progression from the data distribution to the prior, providing a well-defined framework for the reverse process to reconstruct data from noise.

Reverse Process The reverse process reconstructs the original data distribution by inverting the forward process, progressively removing noise to generate data samples:

- **DDPM (Discrete Time):** The reverse process in DDPMs is a Markov chain that progressively denoises the data at each step. The conditional distribution for each step is expressed as:

$$p_{X_{t-1}|X_t,\theta}(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}\left(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t,t), \beta_t^2\mathbf{I}\right),$$

where $\mu_\theta(\mathbf{x}_t,t)$ is a neural network that predicts the mean of the denoised distribution, given the noisy input $\mathbf{x}_t$ and the timestep t. Sampling is performed iteratively by drawing from $p_{X_{t-1}|X_t,\theta}(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$, starting with $\mathbf{x}_T \sim \pi = \mathcal{N}(\mathbf{0},\mathbf{I})$.

- **SGM (Continuous Time):** In SGMs, the reverse process is governed by the reverse-time SDE, derived from Anderson's theorem [2]:

$$dX_t = \left[f(X_t,t) - g(t)^2 \nabla_{X_t} \log p_t(X_t)\right] dt + g(t)\, d\bar{W}_t,$$

where:

– $\nabla_{X_t} \log p_t(X_t)$ is the gradient of the log density (score function) of the perturbed distribution at diffusion time t. In practice, the score function is approximated by a neural network $s_\theta(X_t,t)$, referred to as the score model,

– $\bar{W}_t$ is a reverse-time Wiener process.

Sampling proceeds by solving this reverse-time SDE, starting from $X_T \sim \pi \approx p_T(X_T)$ (e.g., $\mathcal{N}(\mathbf{0},\mathbf{I})$) and integrating backward from $t = T$ to $t = 0$. This iterative process generates high-quality samples that closely follow the original data distribution.

The reverse process lies at the core of diffusion-based generative modeling, enabling the transformation of simple noise into complex data distributions while preserving high fidelity.

Training Objective Training involves learning to approximate the reverse process by minimizing specific loss functions:

- **DDPM:** In the DDPM framework, the training objective is formulated to minimize an expression that serves as an upper bound on the negative log-likelihood of the data. This objective is defined as:

$$\mathscr{L}_{\text{DDPM}}(\theta) = \sum_{t=1}^{T} \mathbb{E}_{x_0 \sim p(X_0),\, \varepsilon_t \sim \mathscr{N}(\mathbf{0},\mathbf{I})} \left[\left\| \varepsilon_t - \varepsilon_\theta \left(\sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \varepsilon_t, t \right) \right\|^2 \right],$$

  where:

  - $\varepsilon_t \sim \mathscr{N}(0, \mathbf{I})$ is Gaussian noise added at time step t,
  - $\varepsilon_\theta(\mathbf{x}_t, t)$ is a neural network that estimates the noise component in $\mathbf{x}_t$,
  - $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$ is the cumulative product of the noise schedule parameters,
  - $\alpha_t = 1 - \beta_t$, and β_t is the variance of the noise added at each time step.

  By minimizing $\mathscr{L}_{\text{DDPM}}(\theta)$, we effectively minimize an upper bound on the negative log-likelihood, enhancing the model's capacity to generate realistic data samples.

  The mean μ_θ of the reverse process used during sampling is derived from the predicted noise ε_θ as follows:

$$\mu_\theta(\mathbf{x}_t, t) = \frac{\sqrt{\bar{\alpha}_{t-1}}\,\beta_t}{1-\bar{\alpha}_t}\,\hat{\mathbf{x}}_0 + \frac{\sqrt{\alpha_t}\,(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\,\mathbf{x}_t,$$

  where:

  - $\hat{\mathbf{x}}_0 = \frac{\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\,\varepsilon_\theta(\mathbf{x}_t, t)}{\sqrt{\bar{\alpha}_t}}$ is the reconstructed noiseless data point.

  This formulation ensures that the reverse diffusion process effectively removes noise at each step, gradually transforming pure noise into high-quality data samples during generation.

- **SGM:** In the Score-Based Generative Modeling (SGM) framework, the neural network s_θ is trained to approximate the score function of the perturbed data distribution $p_t(\mathbf{x})$.

This is achieved by minimizing the weighted **Denoising Score Matching** objective:

$$\mathscr{L}_{\text{DSM}}(\theta) = \frac{1}{2}\int_0^T \lambda(t)\,\mathbb{E}_{\mathbf{x}_0,\mathbf{x}_t\sim P(X_0,X_t)}\left[\left\|s_\theta(\mathbf{x}_t,t) - \nabla_{\mathbf{x}_t}\log p_{X_t|X_0}(\mathbf{x}_t \mid \mathbf{x}_0)\right\|^2\right] dt,$$

where:

- $p_{X_t|X_0}(\mathbf{x}_t \mid \mathbf{x}_0)$ is the perturbation kernel of the forward Stochastic Differential Equation (SDE),
- $\lambda(t)$ is a weighting function,
- $\mathbf{x}_t$ is the diffused data point at time t.

Song et al. [181] showed that for the particular choice $\lambda(t) = g(t)^2$, where $g(t)$ is the diffusion coefficient of the forward SDE, the $\mathscr{L}_{\text{DSM}}$ objective serves as an upper bound on the negative log-likelihood.

In practice, a different weighting function—often referred to as the *simple weighting*—is commonly used:

$$\lambda(t) = \sigma(t)^2,$$

where $\sigma(t)^2$ denotes the variance of the forward perturbation kernel $p_{X_t|X_0}(\mathbf{x}_t \mid \mathbf{x}_0)$. This weighting function has been found to yield superior generative performance, particularly in the image domain where it was initially tested. Consequently, it has become the *de facto* standard for practical implementations.

Equivalence Between DDPM and SGM Denoising Diffusion Probabilistic Models (DDPMs) and Score-Based Generative Models (SGMs) are equivalent in the limit as the number of timesteps $N \to \infty$. Specifically, the forward process in a DDPM converges to the Variance Preserving Stochastic Differential Equation (VP-SDE) used in SGMs:

$$dX_t = -\tfrac{1}{2}\beta(t)X_t\,dt + \sqrt{\beta(t)}\,dW_t,$$

where $\beta(t)$ is the continuous noise schedule and W_t is a standard Wiener process. This result is detailed in Song et al.'s paper *Score-Based Generative Modeling through Stochastic Differential Equations* (2020).

Probability Flow ODE

An alternative to the stochastic reverse-time SDE is the **probability flow ODE**, which provides a deterministic mapping between the data distribution and the prior. The probability

flow ODE is defined as:

$$\frac{d\mathbf{x}_t}{dt} = f(\mathbf{x}_t, t) - \frac{1}{2} g(t)^2 s_\theta(\mathbf{x}_t, t),$$

where:

- $f(\mathbf{x}_t, t)$ and $g(t)$ are the drift and diffusion coefficients of the forward SDE,
- $s_\theta(\mathbf{x}_t, t)$ is the approximation of the score function at time t.

Under the assumption of a perfectly approximated score function, the probability flow ODE shares the same marginal probability distributions as the reverse-time SDE for all diffusion times. This equivalence arises from considering the Fokker-Planck equations corresponding to both processes, which govern the time evolution of probability densities.

Computational Efficiency. In practice, integrating the probability flow ODE has been observed to be computationally more efficient than simulating the reverse-time SDE. Fewer integration steps are typically required to achieve similar sample quality, making it a preferred choice for sampling in diffusion models. This efficiency stems from the deterministic nature of the ODE, which allows for larger step sizes, unlike the stochastic SDE that often requires smaller step sizes to maintain sample fidelity.

Likelihood Computation with the Probability Flow ODE. The probability flow ODE enables likelihood computation via the instantaneous change of variables formula. Given the score function $s_\theta(\mathbf{x}_t, t)$, the log-likelihood of a data point $\mathbf{x}_0$ can be expressed as:

$$\log p_{X_0}(\mathbf{x}_0) = \log p_{X_T}(\mathbf{x}_T) + \int_0^T \nabla_{\mathbf{x}_t} \cdot \mathbf{f}_\theta^{\mathrm{ODE}}(\mathbf{x}_t, t)\, dt,$$

where $\mathbf{f}_\theta^{\mathrm{ODE}}(\mathbf{x}_t, t)$ is the drift term of the probability flow ODE. Direct computation of the divergence term $\nabla_{\mathbf{x}_t} \cdot \mathbf{f}_\theta^{\mathrm{ODE}}(\mathbf{x}_t, t)$ is often computationally expensive. To address this, the Skilling-Hutchinson trace estimator is used:

$$\nabla_{\mathbf{x}_t} \cdot \mathbf{f}_\theta^{\mathrm{ODE}}(\mathbf{x}_t, t) \approx \mathbb{E}_{\varepsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[\varepsilon^\top \nabla_{\mathbf{x}_t} \mathbf{f}_\theta^{\mathrm{ODE}}(\mathbf{x}_t, t) \varepsilon \right],$$

where $\nabla_{\mathbf{x}_t} \mathbf{f}_\theta^{\mathrm{ODE}}(\mathbf{x}_t, t)$ is the Jacobian of $\mathbf{f}_\theta^{\mathrm{ODE}}$. The term inside the expectation can be efficiently computed using Jacobian-vector product calculations, making this approach both practical and scalable for accurate likelihood computation.

Conditional Generation

Diffusion models can be naturally extended to conditional generation tasks, where the objective is to model the conditional distribution $p_{X_0|C}(\mathbf{x}_0 \mid \mathbf{c})$, with $\mathbf{c}$ representing the conditioning variable (e.g., class labels, textual descriptions, or other modalities). This

extension enables generating data conditioned on specific information, with applications such as super-resolution, inpainting, and text-guided generation.

A widely used approach for conditional generation in diffusion models is the weighted **Conditional Denoising Score-Matching** (CDSM) objective. The core idea is to train a neural network $s_\theta(\mathbf{x}_t, \mathbf{c}, t)$ that takes as input the noisy data $\mathbf{x}_t$, the condition $\mathbf{c}$, and the time t, and outputs an estimate of the conditional score $\nabla_{\mathbf{x}_t} \log p_{X_t|C}(\mathbf{x}_t \mid \mathbf{c})$. The training objective is defined as:

$$\mathscr{L}_{\text{CDSM}}(\theta) = \frac{1}{2}\int_0^T \mathbb{E}_{\mathbf{x}_0,\mathbf{x}_t,\mathbf{c}\sim p(X_0,X_t,C)}\left[\lambda(t)\left\|s_\theta(\mathbf{x}_t,\mathbf{c},t) - \nabla_{\mathbf{x}_t}\log p_{X_t|X_0}(\mathbf{x}_t \mid \mathbf{x}_0)\right\|^2\right] dt. \tag{1.1}$$

The only difference between the DSM objective used for unconditional generation and the CDSM objective used for conditional generation lies in the incorporation of the conditioning variable $\mathbf{c}$ into the score model. Note that the regression target—the score function of the perturbation kernel—remains the same. While early empirical results were promising, it was unclear why the minimizer of the CDSM objective approximates the true conditional score function.

In Chapter 2, we provide the first formal proof of consistency for the CDSM estimator, establishing its theoretical validity and confirming its utility in modeling conditional distributions.

Strengths and Weaknesses of the Unified Framework The unified framework of diffusion and score-based generative models exhibits several notable strengths. Among its strengths, the framework ensures stable training by avoiding issues such as mode collapse and instability that commonly afflict adversarial training methods like Generative Adversarial Networks (GANs). Its flexibility allows it to model complex, high-dimensional, and multi-modal data distributions, making it highly adaptable for various applications. Additionally, it is grounded in stochastic process theory with well-defined objectives, providing a solid theoretical foundation that guides both its development and analysis. However, a significant weakness of this framework is its slow sampling speed. Generating samples typically requires hundreds or even thousands of iterative steps, resulting in significant computational costs. This can make the approach impractical for applications where rapid sample generation is crucial.

Advancements in Sampling Speed Significant progress has been made in accelerating the sampling speed of diffusion models, effectively addressing their primary limitation of slow

generation due to iterative steps. A critical observation in this area is that the probability flow ODE, which shares the same marginal distribution as the reverse SDE, is generally preferred for faster sampling. The ODE formulation tends to produce better samples than the corresponding SDE in the low discretization regime [183]. Consequently, many methods that accelerate diffusion models focus on learning to efficiently integrate the probability flow ODE, often as a post-training optimization.

Techniques such as **Denoising Diffusion Implicit Models (DDIM)** [179] reformulate the reverse diffusion process by utilizing non-Markovian dynamics, significantly reducing the number of required sampling steps without compromising sample quality. **Latent Diffusion Models (LDMs)** [164] enhance efficiency by performing diffusion in a compressed latent space, drastically decreasing computational demands while preserving high-resolution outputs. **Progressive Distillation** [171] further compresses the multi-step diffusion process into models capable of generating high-fidelity samples in significantly fewer steps—sometimes even a single step. Recently, **Consistency Models** [180] have been introduced, directly mapping noise to data in one step while retaining the advantages of iterative methods.

Collectively, these advancements have significantly reduced sampling times and made diffusion models practical for real-time and interactive applications across various domains.

1.2 Self-Supervised Learning

1.2.1 Motivation

In today's data-rich environment—spanning social media, sensor outputs, and extensive collections of images, videos and text—the primary challenge is not the lack of data but our ability to utilize it effectively. Despite the vast amounts of information available, much remains unlabeled and underexploited due to the high costs and impracticalities associated with manual labeling. This situation poses a crucial question: how can we unlock the potential of unlabeled data to develop intelligent systems that learn, adapt, and generalize in a manner akin to human learning?

Self-supervised learning (SSL) emerges as a compelling approach to this challenge. Referred to by Yann LeCun as the "dark matter of intelligence" [110], SSL enables machines to learn from the abundant unlabeled data by exploiting the inherent structures within it. LeCun remarks:

> "Common sense helps people learn new skills without requiring massive amounts of teaching for every single task. Self-supervised learning is one of the most

> promising ways to build such background knowledge and approximate a form of common sense in AI systems."

Analogous to how a child learns about the world through observation and pattern recognition rather than explicit instruction, SSL allows models to derive meaningful representations from data without manual labels. The transformative impact of SSL is evident across multiple domains::

Natural Language Processing (NLP): SSL has transformed NLP by enabling models to understand and generate human language with improved contextual awareness. Early models like Word2Vec [130] and GloVe [150] learned semantic relationships between words through context prediction. Advanced models such as BERT [41] and GPT [158] employed masked language modeling to pretrain on extensive text corpora, enhancing capabilities in tasks like text completion and machine translation.

Computer Vision (CV): In CV, SSL techniques like Contrastive Predictive Coding (CPC) [143], SimCLR [33], and MoCo [72] have enabled models to learn visual representations without reliance on labeled datasets. For example, Facebook's SEER model [65], trained on a billion unlabeled images, achieved state-of-the-art performance on ImageNet, illustrating the efficacy of learning from unlabeled visual data.

Speech Recognition: SSL models such as wav2vec [174] and wav2vec 2.0 [9] have significantly advanced speech recognition by learning representations from raw audio data. By predicting future audio samples from past ones, these models capture essential phonetic and linguistic features, improving transcription accuracy and the handling of diverse accents and intonations.

Multimodal Learning: SSL has significantly advanced the integration of multiple data modalities, enabling AI systems to learn joint representations that bridge the gap between domains like vision, language, and audio. Pioneering models such as CLIP [157] have shown remarkable success in aligning images and text within a shared latent space, enabling a wide range of applications. These include image captioning, where the model generates descriptive text for images; text-based image retrieval, allowing users to find images using natural language queries; and zero-shot classification, where the model classifies images into categories it has not seen during training, guided only by textual descriptions.

These advancements are not merely incremental; they signify a fundamental shift in machine learning methodologies. By leveraging unlabeled data, SSL enables models to develop a form of common sense, understanding context and patterns previously inaccessible through supervised learning alone.

The potential applications of SSL are extensive. It opens pathways to developing AI systems capable of understanding and generating human-like language and vision, and

learning new tasks with minimal supervision. Self-supervised learning represents a significant step toward creating AI that learns and adapts more like humans, enhancing intelligence and adaptability in artificial systems.

1.2.2 Mathematical Formulation of SSL

Formally, let $\mathscr{D} = \{\mathbf{x}_i\}_{i=1}^N \subset \mathbb{R}^d$ be a dataset of unlabeled samples drawn from an unknown distribution $P_{\mathbf{x}}$. The objective of self-supervised representation learning is to learn an encoder function $f_\theta : \mathbb{R}^d \to \mathbb{R}^k$, where $k \leq d$, that maps input data to a latent space, producing meaningful representations $\mathbf{h}_i = f_\theta(\mathbf{x}_i)$.

The encoder f_θ is parameterized by θ and is trained by minimizing a self-supervised loss function $\mathscr{L}(\theta; \mathscr{D})$ designed to capture essential features of the data:

$$\theta^* = \arg\min_\theta \mathbb{E}_{\mathbf{x}_i \sim P_{\mathbf{x}}}[\mathscr{L}(\mathbf{x}_i; \theta)].$$

The design of $\mathscr{L}$ depends on the specific self-supervised learning approach and aims to enforce properties such as invariance to data augmentations, ability to predict contextual information, or alignment with semantic structures.

1.2.3 Generative vs. Discriminative SSL

Self-supervised learning methods can be broadly categorized into two paradigms: **generative** and **discriminative** approaches. Understanding the strengths and applications of each is crucial for selecting appropriate methods for specific tasks.

Generative Approaches

These methods aim to extract meaningful latent representations by modeling the underlying data distribution. These approaches can be broadly categorized into two types based on the scope of the distribution being modeled:

1. **Full data distribution** $p(\mathbf{x})$
2. **Masked data distribution** $p(\mathbf{x}_{\textbf{masked}} | \mathbf{x}_{\textbf{context}})$

Learning Latent Representations by Modeling the Full Distribution $p(\mathbf{x})$

- *Auto-Regressive (AR) Models*: AR models factorize the data distribution into a product of conditional probabilities. These models generate each data element sequentially, conditioned on the previous ones.

  - *GPT and GPT-2* [158]: Learn language representations by predicting the next token given preceding context, enabling robust text generation and understanding. GPT embeddings serve as inputs for fine-tuned classifiers in NLP tasks.
  - *PixelRNN and PixelCNN* [142]: Model pixel dependencies in images for self-supervised representation learning, useful for tasks like image completion. PixelCNN representations have been used in image segmentation.

- *Normalizing Flows*: These models transform a simple latent distribution into a complex target distribution using a sequence of invertible mappings, preserving exact likelihoods. Examples include:

  - *FlowGMM* [88]: Combines normalizing flows with a Gaussian mixture model in the latent space to extract representations useful for tasks such as text classification, tabular data analysis, and semi-supervised image classification.

- *Auto-Encoding (AE) Models*: Autoencoders reconstruct data from latent representations, learning robust embeddings for various downstream tasks. Examples include:

  - *Variational Autoencoders (VAEs)* [102]: Introduce probabilistic latent spaces, learning smooth representations useful for clustering and semi-supervised learning.
  - *VQ-VAE* [144]: Extends VAEs to discrete latent spaces, providing meaningful representations that have been widely applied in downstream tasks:
    * In *audio processing*, VQ-VAE representations have been used in *wav2vec* [9] for unsupervised pretraining on speech data, enabling improvements in downstream speech recognition tasks by learning phonetic and linguistic features from raw audio.
    * In *vision*, the latent codes are inputs to autoregressive models for image generation, inpainting, and enhancement tasks [160].
    * In *multimodal learning*, VQ-VAE representations bridge modalities such as text and images, aiding in tasks like image captioning and cross-modal retrieval [45].

- *GAN-Based Models*: Generative Adversarial Networks (GANs) learn latent representations by training a generator and a discriminator adversarially. Recent advancements adapt GANs for representation learning. Examples include:

  - *StyleGAN* [94]: Generates high-quality images with interpretable latent spaces that enable fine-grained editing (e.g., facial expressions and lighting adjustments). StyleGAN embeddings are widely used in image synthesis and manipulation tasks.
  - *VQ-GAN* [51]: Combines vector quantization and adversarial training to learn discrete latent representations. These representations are used for tasks such as text-to-image generation, image compression, and cross-modal retrieval.

Learning Latent Representations by Modeling the Masked Distribution $p(\mathbf{x}_{\text{masked}}|\mathbf{x}_{\text{context}})$

- **Masked Image Modeling:** Models learn representations by reconstructing occluded parts of an image, leveraging context to infer the missing information.
  - *Examples:*
    * **BEiT** [10]: Learns representations for downstream tasks like image classification and segmentation by reconstructing masked patches of images.
    * **Masked Autoencoders (MAE)** [71]: Focuses on learning efficient representations by masking a large portion of the input image and reconstructing it from the unmasked portion.
  - *Downstream Applications:* Representations from masked image models are particularly useful for visual tasks such as object detection, segmentation, and classification, where capturing spatial relationships is critical.
- **Masked Language Modeling:** In NLP, masked language models predict missing words or tokens in a sequence, learning rich contextual embeddings.
  - *Examples:*
    * **BERT** [41]: A foundational model in NLP, pretraining on masked tokens enables fine-tuning for tasks such as sentiment analysis and named entity recognition.
    * **RoBERTa** [121]: Builds on BERT by optimizing training dynamics and is widely used for text classification and question answering.
  - *Downstream Applications:* Masked language model representations are extensively used for text-based tasks, including translation, summarization, and retrieval.

Discriminative Approaches

These methods focus on learning robust feature representations by solving pretext tasks or leveraging contrastive objectives without explicitly modeling the data distribution. The main motivation is to learn representations that are useful for downstream tasks by capturing essential features and invariances in the data. Key techniques include:

- **Contrastive Learning**:
  - *Motivation and Logic*: Contrastive learning aims to learn robust feature representations by maximizing agreement between similar pairs (positive pairs) and minimizing agreement between dissimilar pairs (negative pairs). The central idea is to structure the latent space such that representations of similar samples are closer, while representations of different samples are farther apart.
  - *Generalized Contrastive Loss*: [201] unified contrastive losses under a general framework, where the loss is expressed as:

$$\mathscr{L}_{\phi,\psi}(\theta) = \sum_{i=1}^{N} \phi\left(\sum_{j\neq i} \psi\left(\|z_i - z_j\|_2^2 - \|z_i - z_i'\|_2^2\right)\right),$$

    where:
    * z_i and z_i' are the representations of a sample i and its positive pair (augmented view of the same sample).
    * $\|z_i - z_j\|_2^2$ is the squared distance between the representations of sample i and a negative sample j.
    * ϕ and ψ are monotonically increasing and differentiable scalar functions that control how positive and negative distances are weighted.

    Specific choices of ϕ and ψ lead to well-known contrastive learning methods:
    * **InfoNCE Loss** [143]: Setting $\phi(x) = \tau\log(\varepsilon + x)$ and $\psi(x) = \exp(x/\tau)$ results in:

$$\mathscr{L}_{\text{InfoNCE}} = -\tau\sum_{i=1}^{N} \log \frac{\exp\left(-\|z_i - z_i'\|_2^2/\tau\right)}{\varepsilon\exp\left(-\|z_i - z_i'\|_2^2/\tau\right) + \sum_{j\neq i}\exp\left(-\|z_i - z_j\|_2^2/\tau\right)}.$$

    * **SimCLR Loss** [33]: Simplifies InfoNCE by setting $\varepsilon = 0$, removing the stabilization term in the denominator.

- **Self-Distillation**:

- *Motivation and Logic*: These methods aim to learn representations without the need for negative samples, addressing issues like the need for large batch sizes or memory banks in contrastive learning. The model learns by predicting its own representations under different augmentations, promoting consistency.
- *Training Objective*: The model minimizes a prediction loss between representations of different augmented views. For instance, in BYOL, the loss is:

$$\mathscr{L}_{\text{BYOL}} = \left\| \mathbf{q}_i - \text{stopgrad}(\mathbf{z}_i^+) \right\|_2^2,$$

  where:
  - $\mathbf{q}_i = f_\theta(\mathbf{x}_i)$ is the output of the online network.
  - $\mathbf{z}_i^+ = f_{\theta'}(\mathbf{x}_i^+)$ is the output of the target network.
  - stopgrad indicates that gradients are not backpropagated through this path.
  - θ' is an exponential moving average of the parameters θ.
- *Examples*:
  - *BYOL* [67]: Uses an online and a target network to predict one view from another, with the target network updated via an exponential moving average.
  - *SimSiam* [34]: Simplifies BYOL by removing the momentum encoder and employing a stop-gradient operation to prevent representational collapse.

- **Feature Decorrelation Methods**:
  - *Motivation and Logic*: Feature decorrelation methods aim to learn rich and diverse representations by reducing redundancy among feature dimensions. Inspired by Canonical Correlation Analysis (CCA), these approaches encourage each component of the embedding to capture unique information, promoting uncorrelated and informative features. By focusing on decorrelating feature dimensions, these methods prevent representational collapse and enhance the quality of learned representations without relying on negative samples or complex training schemes.
  - *Training Objective*: The central idea is to make embeddings from different augmented views of the same data similar (invariance) while ensuring that different feature dimensions are uncorrelated (decorrelation). This is achieved by combining an invariance term with a decorrelation term in the loss function, encouraging both similarity across views and diversity among features.
    - *Barlow Twins* [220] employs two identical networks to process different augmented views of the same image. It computes a cross-correlation matrix

$\mathbf{C}$ between the embeddings from the two networks. The loss function encourages the diagonal elements of $\mathbf{C}$ to be close to one (promoting invariance) and the off-diagonal elements to be close to zero (reducing redundancy):

$$\mathscr{L}_{\text{BT}} = \sum_{i=1}^{d} (1 - \mathbf{C}_{ii})^2 + \lambda \sum_{i \neq j} \mathbf{C}_{ij}^2.$$

This approach enables the learning of features that are invariant to augmentations while ensuring each dimension captures distinct information.

* *VICReg* [11] introduces a loss function with three components. The invariance term minimizes the mean squared error between embeddings from different views. The variance term ensures the standard deviation of each feature dimension exceeds a threshold to prevent collapse. The covariance term penalizes off-diagonal covariance elements to reduce redundancy. The loss is expressed as:

$$\mathscr{L}_{\text{VICReg}} = \underbrace{\frac{1}{B} \sum_{b=1}^{B} \|\mathbf{z}_b^{(1)} - \mathbf{z}_b^{(2)}\|_2^2}_{\text{Invariance}} + \gamma \underbrace{\sum_{j=1}^{d} \max(0, s - \sigma(\mathbf{z})_j)}_{\text{Variance}} + \mu \underbrace{\sum_{i \neq j} \text{Cov}(\mathbf{z})_{ij}^2}_{\text{Covariance}},$$

where γ and μ balance the terms, s is the variance threshold, and $\text{Cov}(\mathbf{z})_{ij}$ is the off-diagonal element of the covariance matrix of $\mathbf{z}$.

- **Clustering-Based Methods**:
  - *Motivation and Logic*: These approaches leverage unsupervised clustering to group similar data points, learning representations that capture group-level semantic structures. The model alternates between clustering the data in the representation space and updating the encoder to produce features that align with these clusters.
  - *Training Objective*: The encoder is trained to predict pseudo-labels derived from cluster assignments. The loss function typically involves cross-entropy between the predicted labels and the pseudo-labels:

$$\mathscr{L}_{\text{cluster}} = \sum_{i=1}^{N} \ell_{\text{CE}}(f_\theta(\mathbf{x}_i), y_i),$$

where y_i is the cluster assignment for $\mathbf{x}_i$ obtained via a clustering algorithm (e.g., k-means), and ℓ_{CE} denotes the cross-entropy loss.

  - *Examples*:
    - *DeepCluster* [28]: Performs k-means clustering on the learned features and uses the cluster assignments as pseudo-labels for training.
    - *SwAV* [29]: SwAV uses siamese networks to generate embeddings for two augmented views of an image and aligns them with trainable prototypes. It computes cluster assignments online using the Sinkhorn-Knopp algorithm and minimizes cross-entropy between these assignments and predicted probabilities, enabling efficient training without pairwise comparisons.

- **Pretext Task-Based Methods**
  - *Motivation and Logic*: Pretext task-based methods aim to learn useful representations by training models on auxiliary tasks where the labels can be derived automatically from the data itself. These tasks, often unrelated to the downstream tasks, encourage the model to capture meaningful patterns and structures in the data that generalize well across applications. By solving these simple yet informative tasks, the model learns features that are transferable to various downstream tasks without requiring manual annotations.
  - *Examples*:
    - **Rotation Prediction** [58]: The model learns representations by predicting the rotation angle applied to an image. Given an image rotated by one of four angles (0°, 90°, 180°, or 270°), the model classifies the rotation angle. This task encourages the model to understand spatial relationships and object structures within the image.
    - **Jigsaw Puzzle Solving** [137]: The model is tasked with solving jigsaw puzzles by predicting the correct arrangement of shuffled image patches. This pretext task forces the model to learn contextual and semantic relationships between different parts of an image.
    - **Temporal Ordering** [132]: For video data, the model learns representations by predicting the correct temporal order of video frames. This task helps the model capture temporal dependencies and motion dynamics in videos.

Choosing Between Generative and Discriminative SSL

The choice between generative and discriminative self-supervised learning methods depends on the specific objectives and requirements of the task. Each approach has its strengths and is more suitable under certain circumstances.

Generative SSL is more useful when: Generative SSL methods excel in scenarios that demand a fine-grained understanding of the data distribution. These methods capture detailed and nuanced information within the extracted representations, making them ideal for tasks that require precise and intricate information about the data. For instance, in *image editing and manipulation*, generative models such as StyleGAN [94, 95] enable sophisticated modifications by navigating the latent space. By solving optimization problems to move in specific directions within this space, attributes like facial expressions, styles, or lighting conditions can be altered, providing a level of fine-grained control that discriminative models typically cannot offer. Additionally, generative SSL is advantageous in applications like anomaly detection, where understanding the complete data distribution allows for the identification of outliers, and in cross-modal tasks such as text-to-image generation, where translating between different data modalities requires comprehensive distribution modeling.

Discriminative SSL is more useful when: Discriminative SSL methods are preferable when the primary goal is to learn efficient, task-relevant representations tailored for specific downstream applications. These methods optimize embeddings to effectively distinguish between different classes or instances, making them highly suitable for tasks such as classification and retrieval. For example, in image classification, discriminative models like SimCLR [33] and MoCo [72] produce embeddings that enhance performance by focusing on feature discrimination. Furthermore, discriminative SSL is advantageous in environments with limited computational resources, as these methods often require fewer resources and are more scalable. They are also ideal for real-time or latency-sensitive applications, where fast inference and low latency are critical, such as online recommendation systems or real-time monitoring.

Blurring the Lines: Hybrid Approaches

The distinction between generative and discriminative SSL methods is increasingly becoming blurred, with hybrid approaches emerging that combine the strengths of both paradigms. These hybrid models leverage the fine-grained distribution modeling of generative methods alongside the efficient, task-specific representation learning of discriminative methods. No-

table examples include BEiT [10] and iBOT [225], which integrate masked image modeling (a generative task) with self-distillation objectives (a discriminative task). By doing so, these models capture rich data representations while maintaining computational efficiency, effectively addressing a broader range of tasks.

1.2.4 Utilizing Learned Representations in Downstream Tasks

Once the representation function f_θ is learned, the representations $\mathbf{h}_i = f_\theta(\mathbf{x}_i)$ can be utilized in various downstream tasks. These representations capture essential features and structures in the data, making them valuable inputs for different applications.

Classification

In classification problems, the goal is to assign input data to one of several predefined categories. The learned representations $\mathbf{h}_i$ serve as informative features that can be used to train a classifier. Specifically, we can define a classifier $g_\phi : \mathbb{R}^k \to \mathbb{R}^C$, where C is the number of classes, and ϕ represents the parameters of the classifier (e.g., weights of a fully connected layer).

The classifier is trained on a labeled dataset $\{(\mathbf{x}_j, y_j)\}_{j=1}^M$, where $y_j \in \{1, 2, \ldots, C\}$ is the class label for sample $\mathbf{x}_j$. The training involves minimizing a supervised loss function, typically the cross-entropy loss:

$$\phi^* = \arg\min_\phi \frac{1}{M} \sum_{j=1}^{M} \ell_{\text{CE}}(g_\phi(\mathbf{h}_j), y_j),$$

where $\mathbf{h}_j = f_\theta(\mathbf{x}_j)$ and ℓ_{CE} denotes the cross-entropy loss function.

The representations $\mathbf{h}_i$ often capture high-level semantic features and are invariant to variations in the input data that are irrelevant to the classification task, such as changes in lighting, orientation, or background. This invariance leads to better generalization and robustness, improving classification performance even when labeled data is limited.

In practice, the representations can be utilized in two ways:

- **Fixed Feature Extraction**: The encoder f_θ is kept frozen, and only the classifier g_ϕ is trained on the labeled data. This approach is useful when computational resources are limited or when the labeled dataset is small.

- **Fine-Tuning**: Both the encoder f_θ and the classifier g_ϕ are trained (or fine-tuned) jointly on the labeled data. This allows the model to adapt the learned representations to the specific nuances of the downstream task, often leading to better performance.

Other Downstream Tasks

Similar logic can be applied to other downstream tasks such as object detection, semantic segmentation, instance segmentation, image captioning, and more. In these tasks, the extracted representations $\mathbf{h}_i$ serve as inputs to task-specific prediction heads. For example, in object detection, the representations can be used to extract feature maps that help in localizing and classifying objects within images. In semantic segmentation, the representations provide spatial and contextual information necessary for assigning class labels to each pixel in an image. For image captioning, the representations capture visual features that can be input to sequence models to generate descriptive captions for images.

In all these tasks, the pretrained encoder f_θ acts as a feature extractor, and the task-specific prediction heads are trained to perform the desired task using the representations as input. Similar to classification, the encoder f_θ can be kept fixed or fine-tuned along with the prediction heads. This flexibility allows the model to either leverage the general representations learned during self-supervised pretraining or adapt them to the specific nuances of the downstream task. This approach leverages the strengths of self-supervised learning to enhance performance across various domains and applications.

1.2.5 Challenges and Limitations in Current Methodologies

Despite significant advancements, current self-supervised learning (SSL) methods face several challenges that hinder their ability to fully capture the underlying structure and complexity of data. In this section, we discuss key weaknesses—particularly those we aim to address in this work—including intrinsic dimension estimation, limited model expressiveness in variational autoencoders (VAEs), and the lack of geometric understanding and interpretability in the latent space.

Intrinsic Dimension Estimation

While SSL methods strive to extract meaningful representations from data, they often do not make an informed choice about the latent dimensionality of these representations. Standard practice involves selecting a latent dimension based on convention or precedent in the literature—commonly using values like 512—regardless of the dataset's inherent complexity. This arbitrary selection can lead to suboptimal performance.

If the optimal latent dimension is k and a significantly larger dimension is chosen, the model may incorporate redundant information in downstream tasks. This redundancy might adversely affect generalization. Conversely, selecting a latent dimension smaller than k risks

losing essential information, leading to reduced performance due to insufficient representation capacity.

Developing methods that can estimate or inform the choice of the intrinsic dimensionality of the data manifold is crucial. Such methods enable models to learn representations that are both efficient and effective, capturing the essential structure of the data without unnecessary complexity.

In Chapter 3, we present a novel method for estimating the latent dimension using diffusion models trained on the data distribution. We theoretically and experimentally demonstrate that diffusion (or score-based) models inherently encode information about the intrinsic dimensionality. Specifically, we prove that the Jacobian of the score function—modeled by a diffusion or score-based model—captures the low-dimensional intrinsic structure of the data. When data concentrates around a lower-dimensional manifold of dimension k embedded in a higher-dimensional ambient space of dimension d, the singular value decomposition (SVD) of the Jacobian of the score function yields k vanishing singular values. This finding provides a principled approach to estimating the intrinsic dimension k of the data manifold.

Limited Model Expressiveness in Variational Autoencoders

Variational autoencoders are widely used in unsupervised representation learning due to their ability to model complex data distributions. However, standard VAEs suffer from limited model expressiveness, primarily attributed to two key assumptions: the encoding distribution (often assumed to be a unimodal Gaussian) and the decoding distribution (also assumed to be a unimodal Gaussian).

The assumption of a Gaussian decoding distribution can lead to blurry reconstructions, especially when modeling complex distributions such as natural images. This limitation arises because a unimodal Gaussian cannot capture the multimodal and intricate structures present in such data. As a result, the learned representations may lack the fine-grained details necessary for high-quality reconstructions and downstream tasks that require precise information.

In Chapter 4, we introduce a novel adaptation of the VAE framework called *ScoreVAE*, which addresses this issue by combining a diffusion-time-dependent encoder with an unconditional diffusion model. By employing Bayes' rule for score functions, we analytically derive a robust and flexible model for the reconstruction distribution. Our approach bypasses the unrealistic Gaussian assumption, resulting in a more expressive VAE capable of capturing the complex structures inherent in high-dimensional data like images.

Geometric Understanding and Interpretability of the Latent Space

Another significant limitation of current methodologies is the lack of geometric understanding and interpretability in the latent space. Existing methods often fail to effectively exploit the geometry of the data manifold, leading to representations that may not reflect the true relationships and structures within the data. This lack of interpretability hinders our ability to understand and manipulate the latent representations meaningfully.

In Chapter 5, we introduce a pullback Riemannian metric induced by the score function, capturing the intrinsic dimensionality and geometry of the data manifold. This enables closed-form geodesics, interpretable autoencoding, and a deeper understanding of the manifold's structure. By adapting the normalizing flow framework with isometry regularization and base distribution anisotropy, our method preserves local manifold properties, facilitates dimensionality reduction, and enhances the interpretability of latent space representations.

Chapter 2

Thesis Outline and Contributions

2.1 CAFLOW: Conditional Autoregressive Flows

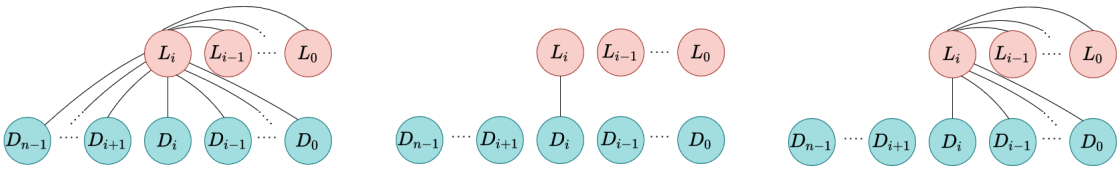


Fig. 2.1 From left to right: ideal dependencies in the i^{th} autoregressive component. Dual-Glow modeling assumption [193]; information is exchanged only between latent spaces having the same dimension. Our modeling assumption; we retain the dependencies between L_i and the latent spaces of lower dimension.

In this chapter, we introduce CAFLOW, a new diverse image-to-image translation model that simultaneously leverages the power of auto-regressive modeling and the modeling efficiency of conditional normalizing flows. We transform the conditioning image into a sequence of latent encodings using a multi-scale normalizing flow and repeat the process for the conditioned image. We model the conditional distribution of the latent encodings by modeling the auto-regressive distributions with an efficient multi-scale normalizing flow, where each conditioning factor affects image synthesis at its respective resolution scale. Our proposed framework performs well on a range of image-to-image translation tasks. It outperforms former designs of conditional flows because of its expressive auto-regressive structure.

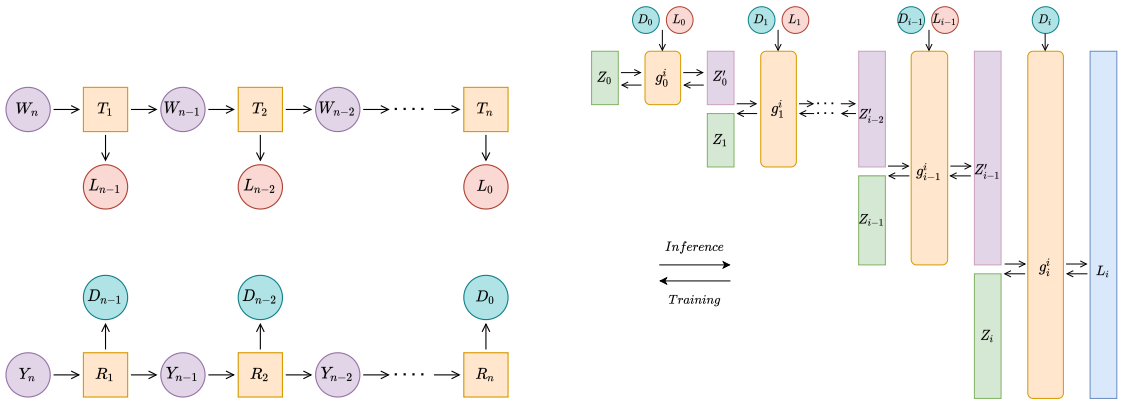


Fig. 2.2 Left: unconditional normalizing flow architecture used to encode conditioning and conditioned images, denoted by $Y_n = Y$ and $W_n = W$ respectively, into a sequence of hierarchical latent variables. Right: design of the conditional transformation G_i^θ that models the i^{th} autoregressive component. The index of the flow i is omitted in both the transformed latent variable Z_j and the intermediate latent variables Z'_j for simplicity.

Originality and Author's Contributions

This chapter is adapted from our published work [12] in the Foundations of Data Science (FoDS) journal. The authors' contributions to this work are as follows:

- **Formulation of Ideas**: The formulation of the CAFLOW framework was entirely my contribution.
- **Experimental Design**: I independently designed all experiments to validate the performance of the proposed framework.
- **Code and Experimental Implementation**: I implemented the entire codebase for the model and conducted all experiments. Christian Etmann provided critical feedback and domain expertise on the training and inference of Normalizing Flows, significantly enhancing the quality of the work.
- **Theory**: I developed the theoretical foundations underpinning the proposed framework, with valuable contributions from Marcelo Carioni, who helped refine and extend the theoretical arguments.
- **Presentation**: Myself and Marcelo Carioni contributed equally to the write-up of this work.

This project was conducted under the supervision of Carola-Bibiane Schönlieb, Christian Etmann, Soroosh Afyouni, and Zoe Kourtzi.

2.2 Non-Uniform Diffusion Models

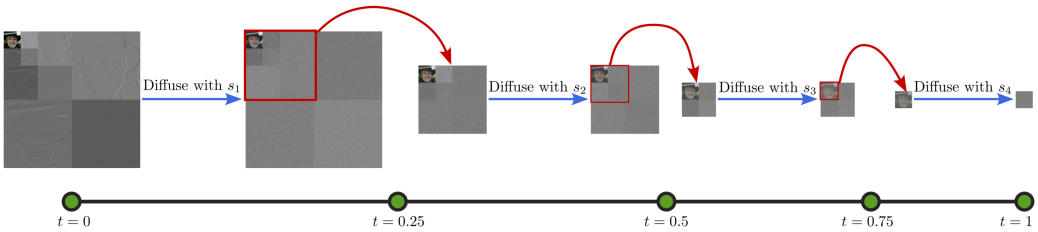


Fig. 2.3 Illustration of a multi-scale diffusion model with three scales. Inspired by multi-scale normalizing flows, this approach diffuses different parts of the image tensor (transformed into multi-level Haar coefficients) at varying speeds. High-frequency detail coefficients diffuse progressively faster, with d_1 diffusing faster than d_2, d_2 faster than d_3, and so on, ensuring that all coefficients reach the same (very low) signal-to-noise ratio (SNR) at their respective terminal diffusion times: $t = 0.25, 0.5, 0.75$, and 1.0, respectively. The low-frequency approximation coefficients a_3 diffuse the slowest, completing their diffusion at $t = 1.0$. The multi-scale structure reduces the dimensionality of the diffusing tensor at each scale, enabling faster computation. Separate neural networks S_1, S_2, S_3, S_4 approximate the score functions at different intervals, leveraging the reduced dimensionality of the intermediate distributions. This hierarchical design mirrors the structure of multi-scale normalizing flows, improving training and sampling efficiency while maintaining high image generation quality.

In this section, we introduce *non-uniform diffusion models*. Unlike standard diffusion approaches that apply the same noise injection schedule to every pixel, non-uniform diffusion models allow different pixels (or groups of pixels) to evolve at varying speeds. This flexibility mirrors the hierarchical approach of multi-scale normalizing flows, where transformations occur at multiple scales, enabling the model to capture image structure more efficiently and produce higher-quality samples in less time. By carefully choosing which parts of the image diffuse faster, non-uniform diffusion opens the door to significantly improved performance, both in terms of image fidelity and computational speed.

We demonstrate that non-uniform diffusion models outperform standard uniform diffusion models by achieving superior FID scores in less training time. Furthermore, these models exhibit remarkable efficiency, sampling up to 4.4 times faster at a resolution of 128×128, with even greater speed-ups anticipated at higher resolutions. Leveraging the adaptability of non-uniform diffusion, we introduce the Conditional Multi-Speed Diffusive Estimator (CMDE), a novel approach derived from a specific choice of non-uniform diffusion. CMDE unifies existing methods for conditional score estimation and delivers performance on par with the widely adopted conditional denoising estimator.

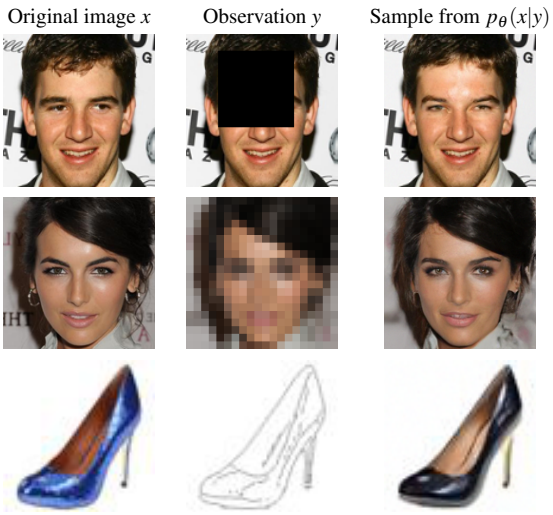


Fig. 2.4 Results from our conditional multi-speed diffusive estimator.

On the theoretical side, we introduce a principled objective for training non-uniform diffusion models and provide a proof of consistency for the conditional denoising estimator, thereby establishing the reliability of the most widely adopted approach to training conditional diffusion models.

Beyond these theoretical and methodological advances, we conduct a comprehensive empirical evaluation of different approaches to training conditional diffusion methods, comparing their effectiveness across tasks such as super-resolution, inpainting, and edge-to-image translation. Finally, to encourage future research and practical adoption, we release MSDiff, an open-source library dedicated to non-uniform and conditional diffusion models, enabling other researchers and practitioners to easily experiment with and build upon our work.

Originality and Author's Contributions

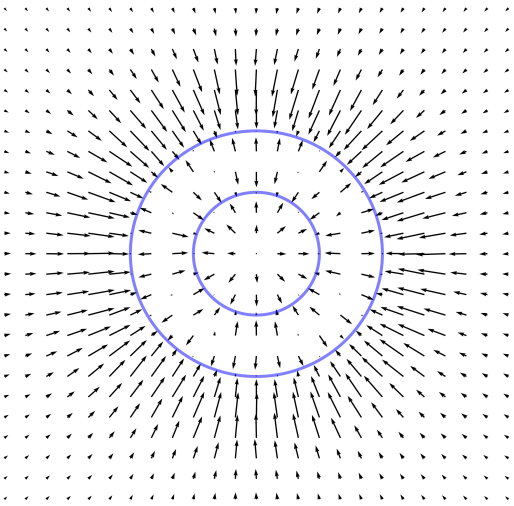
This chapter is adapted from [14]. The author's contributions to this work are as follows:

- **Formulation of Ideas**: I came up with the idea of non-uniform diffusion and how to train non-uniform diffusion models. This idea led to multi-scale diffusion and the CMDE estimator which can be used for training conditional diffusion models. Jan Stanczuk noticed that CMDE is an interpolation between CDE and CdiffE.
- **Experimental Design**: Myself and Jan Stanczuk had equal contribution in the experimental design.

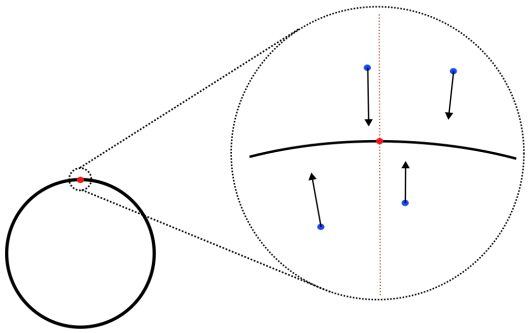
- **Code and Experimental Implementation**: I implemented the experiments presented in the paper. Myself and Jan Stanczuk had equal contribution in the delevopment of the codebase.
- **Theory**: The ideas for the proofs of Theorem 1 and 2 were found together. Final technical ideas were completed by Jan Stanczuk. Jan Stanczuk also compiled the proof of Theorem 3.
- **Presentation**: The design of the paper was discussed together. Jan Stanczuk led the write up of the paper. I contributed to experimental sections of the write up.

This project was conducted under the supervision of Carola-Bibiane Schönlieb and Christian Etmann.

2.3 Diffusion Models Encode the Intrinsic Dimension of Data Manifolds



(a) The data manifold (in blue) and the neural approximation of the score field $\nabla_{\mathbf{x}} \ln p_{t_0}(\mathbf{x})$ obtained from a diffusion model. Near the manifold, the score field is perpendicular to the manifold surface.



(b) The red dot shows a point $\mathbf{x}_0$ on the data manifold where we wish to estimate the dimension. We sample K blue points $\mathbf{x}_t^{(i)}$ in a close neighborhood of the red point and evaluate the score field. The resulting vectors $s_\theta(\mathbf{x}_\varepsilon^{(i)}, \varepsilon)$ point in the normal direction. We put the vectors into a matrix and perform SVD to detect the dimension of the normal space. The dimension of the manifold equals the number of (almost) vanishing singular values.

Fig. 2.5 (Left) Visualization of the score field near the data manifold. (Right) Visualisation of the estimation of the manifold dimension using the trained diffusion model.

In this chapter, we provide a mathematical proof that diffusion models encode data manifolds by approximating their normal bundles. Based on this observation we propose a novel method for extracting the intrinsic dimension of the data manifold from a trained diffusion model. Our insights are based on the fact that a diffusion model approximates the score function i.e. the gradient of the log density of a noise-corrupted version of the target distribution for varying levels of corruption. We prove that as the level of corruption decreases, the score function points towards the manifold, as this direction becomes the direction of maximal likelihood increase. Therefore, at low noise levels, the diffusion model provides us with an approximation of the manifold's normal bundle, allowing for an estimation of the manifold's intrinsic dimension. To the best of our knowledge our method is the first estimator of intrinsic dimension based on diffusion models and it outperforms well established estimators in controlled experiments on both Euclidean and image data.

Originality and Author's Contributions

This chapter is adapted from our published work presented at ICML 2024 on intrinsic dimension estimation with diffusion models [190]. The authors' contributions to this work are as follows:

- **Formulation of Ideas**: Myself and Jan Stanczuk equally contributed to the formulation of ideas.
- **Experimental Design**: Myself and Jan Stanczuk shared equal contributions to the experimental design.
- **Code and Experimental Implementation**: Myself and Jan Stanczuk had equal contribution in the developement of the code framework used to run the experiments. I implemented the following experiments: synthetic image manifolds and MNIST dimensionality estimation. Jan Stanczuk implemented the following experiments: k-spheres, line manifold, spaghetti line, comparison with auto-encoder on MNIST, robustness analysis, and all benchmark methods.
- **Theory**: Jan Stanczuk derived the theoretical results with help from Teo Deveney.
- **Presentation**: Jan Stanczuk and I equally contributed to the write-up, with Teo Deveney contributing to the theoretical sections.

This project was conducted under the supervision of Carola-Bibiane Schönlieb.

2.4 Variational Diffusion Auto-encoder: Latent Space Extraction from Pre-Trained Diffusion Models

Fig. 2.6 Comparison of original and reconstructed images on the FFHQ dataset using our ScoreVAE framework. The left panel presents the original images from the FFHQ dataset, while the right panel displays the corresponding reconstructions generated by ScoreVAE. The results highlight the effectiveness of ScoreVAE in capturing intricate details and preserving high fidelity, overcoming the limitations of traditional VAE models.

In this paper, we introduce **ScoreVAE**, a novel approach that advances the Variational Autoencoder (VAE) framework by addressing fundamental limitations of conventional VAEs. Traditional VAEs model the reconstruction distribution $p(\mathbf{x}|\mathbf{z})$ as a Gaussian, which often leads to overly smoothed and blurry reconstructions. This limitation arises because the Gaussian assumption fails to capture the complexity and multimodality of real-world data distributions, making it difficult for the model to accurately represent intricate details and sharp features. ScoreVAE addresses this issue by combining a diffusion-time-dependent encoder and an unconditional diffusion model. By employing Bayes' rule for score functions, we analytically derive a robust and flexible model for reconstruction distribution $p(\mathbf{x}|\mathbf{z})$. Our approach bypasses the unrealistic Gaussian assumption, resulting in significantly improved image reconstruction quality.

The ScoreVAE framework also simplifies the training dynamics by decoupling the training of the diffusion model and the encoder. This decoupling enables the use of powerful pre-trained diffusion models that can be readily updated or swapped without retraining the entire system. By separating the prior (diffusion model) from the encoder, ScoreVAE achieves higher fidelity reconstructions compared to traditional VAEs and diffusion decoders.

Our experiments on the CIFAR10 and CelebA datasets demonstrate ScoreVAE's superiority in producing sharper images and reducing reconstruction error. These results underscore the practical advantages of ScoreVAE in handling complex, high-dimensional data, and highlight its potential for improved representation learning and controllable generative modeling.

Originality and Author's Contributions

This chapter is adapted from [13]. The authors' contributions to this work are as follows:

- **Formulation of Ideas**: Myself and Jan Stanczuk contributed equally to the formulation of ideas.
- **Experimental Design**: Myself and Jan Stanczuk had equal contributions to the experimental design.
- **Code and Experimental Implementation**: I implemented the code for ScoreVAE model (encoder-only, corrector), adaptation of discrete prior models to continuous time framework and the following experiments: reconstructions with score-VAE and Diffusion Decoder. Jan Stanczuk implemented the beta-VAE and code for semantic manipulation.
- **Theory**: I suggested the sketch of the proof that entailed connecting the ScoreVAE loss to marginal likelihood [181], and Jan Stanczuk derived the rigorous proof.
- **Presentation**: Myself and Jan Stanczuk contributed equally to the final write-up.

This project was supervised by Carola-Bibiane Schönlieb.

2.5 Score-Based Pullback Riemannian Geometry

In this work, we introduce a score-based pullback Riemannian metric that encodes the intrinsic dimensionality and geometry of data under certain distributional assumptions. We show that this data-driven metric can be constructed in practice by modifying normalizing flows with anisotropic base distribution and isometry regularization. This approach yields a scalable framework for computing manifold maps—such as geodesics, exponential maps, distances, and curvature—in closed form. Building on this metric, we additionally construct a Riemannian Auto-encoder (RAE) that recovers the true manifold dimension, offers a global chart of the manifold, and yields an interpretable latent representation thanks to isometry regularization.

(a) Hemisphere (2,3) (b) Sinusoid (1,3) (c) Sinusoid (2,3)

Fig. 2.7 Approximate data manifolds learned by the Riemannian autoencoder generated by score-based pullback Riemannian geometry for three datasets. The orange surfaces represent the manifolds learned by the model, while the blue points correspond to the training data. Each manifold provides a convincing low-dimensional representation of the data, isometric to its respective latent space.

Originality and Author's Contributions

This chapter is adapted from the published work on score-based pullback Riemannian geometry.

- **Formulation of Ideas**: Willem Diepeveen proposed the idea of extracting the data manifold's geometry using a score-based pullback metric. He developed the theoretical motivation of using the score-based metric and showed that the constructed metric allows the computation of manifold maps such as geodesics, exponential map, logarithmic map and distance in closed form. Moreover, he showed that the trained normalizing flow used for the construction of the pullback metric is a Riemannian Auto-encoder.

  I figured out that the proposed metric can be constructed in practice by adapting the framework of Normalizing flows with base distribution anisotropy and l_2 regularisation. Moreover, I discovered the existence of a hessian vector product term that can negatively impact the performance of the Riemannian Auto-encoder if not taken into consideration. Including the hessian vector product term in the regularisation allowed us to use non-affine flows reliably which led to the improvement of the scalability and performance of the method.

- **Experimental Design**: I led the experimental design, incorporating suggestions from Willem Diepeveen and Zakhar Shumaylov.

- **Code and Experimental Implementation**: I implemented the training code and conducted all hyperparameter tuning and experiments. Willem implemented the

functionality for constructing the data-driven manifold maps and the Riemannian autoencoder from a trained model.

- **Theory**: Willem developed the theoretical results. I contributed by explaining the necessity of Hessian-vector product regularization.
- **Presentation**: Zakhar and Willem wrote all sections of the paper except for the experimental section, which I authored.

This project was conducted under the supervision of Carola-Bibiane Schönlieb.

Chapter 3

CAFLOW: Conditional Autoregressive Flows

In this work, we introduce CAFLOW, a new diverse image-to-image translation model that simultaneously leverages the power of autoregressive modeling and the modeling efficiency of conditional normalizing flows. We transform the conditioning image into a sequence of latent encodings using a multi-scale normalizing flow and repeat the process for the conditioned image. We model the conditional distribution of the latent encodings by modeling the autoregressive distributions with an efficient multi-scale normalizing flow, where each conditioning factor affects image synthesis at its respective resolution scale. Our proposed framework performs well on a range of image-to-image translation tasks. It outperforms former designs of conditional flows because of its expressive autoregressive structure.

3.1 Introduction

Generative modeling has emerged as one of the most widely-researched areas in deep learning over the last few years. While generative adversarial networks (GANs) [61] produce state-of-the-art results for images [205], they do not allow for the estimation of likelihoods. Other types of generative methods, however, admit the explicit evaluation of likelihoods of points under the given model, and thus allow for training as maximum likelihood estimators. Normalizing flows [161] (and their continuous formulation, via NeuralODEs [32]) are one such example. These *flow-based* models are diffeomorphic neural networks, which are trained to invertibly transform data into e.g. a normal distribution. This is made possible by a change-of-variables formula, which expresses the likelihood of a data sample in terms of

the normal distribution. Samples from the approximated distribution are then generated by passing samples from a normal distribution through the inverse function.

A straightforward extension of likelihood estimators are *conditional* likelihood estimators, which represent the likelihood of a random variable conditioned on another random variable. This is in particular used in autoregressive models and their flow-based variants (*autoregressive flows* [83]). These types of models explicitly parametrize joint likelihoods via the product rule for probability densities, and can hence be used for creating expressive likelihood estimators, at the cost of more involved computations. This approach is in particular employed by Wavelet Flows [219], which use a hierarchical multi-scale representation via Wavelet decompositions in order to parametrize expressive conditional likelihood functions for images.

Conditional likelihood estimators are also used for domain transfer and image-to-image translation [193, 127, 4, 125] achieving comparable performances to GAN-based models in super-resolution [127, 120], inpainting [125] and image-to-image translation [156, 193, 68]. In domain transfer, the task is to *transfer* a data point from some *origin* domain (e.g. black-and-white images) to a different *target* domain, e.g. color images in this example. The technique of conditioning lends itself well to domain transfer, as it is possible to condition likelihood functions over the target domain on points from the origin domain, thereby making conditional generation possible. In flow-based models, this has been introduced in the form of conditional flows, which function based on this general principle. The most successful examples of conditional flows are [127] and [120] which achieved state-of-the-art performance in image super-resolution by adapting conditional flows to the task of image super-resolution. Both methods rely on powerful feature extractors specifically optimised for image super-resolution to extract conditional features from the low resolution image. This makes them not easily applicable to general inverse problem solving.

Outline of the main contributions. In this work, we introduce CAFLOW, *conditional autoregressive flow*, which combines the idea of conditional flows with the expressiveness of autoregressive models using hierarchical multi-scale flows for domain transfer. Our goal is to improve the performance of conditional flows on general inverse problems in vision.

As a first step, we encode the condition and the target images using invertible networks into a hierarchical sequence of n latent spaces. Then, using the chain rule of probability, we decompose the conditional distribution of the latent encodings into n autoregressive component distributions with the aim of modeling each distribution using a conditional normalizing flow. Modeling the theoretical autoregressive components can be computationally very expensive. For this reason, we used a small set of assumptions that allowed us to design a computationally efficient multiscale normalizing flow at a small modeling flexibility cost.

By carefully designing such a flow based on weak assumptions on the mutual dependencies of the latent encodings, our model allows for exchange of information between latent spaces of different dimension. In particular, our designed architecture is able to capture correlations between different scales, improving on the expressivity of other conditional flow-based models used for domain transfer tasks. The training of the n conditional flows can be parallelized, however, the sampling process is sequential and could become computationally expensive if the number of scales is large. However, in practice, the choice of the invertible networks that map the images to latent encodings does not allow the number of scales n to be large, which makes the sampling time of CAFLOW much smaller than that of pure autoregressive models and slightly greater than that of standard conditional flows.

Our modeling choices are corroborated by strong experimental evidence. Using an ablation study, we confirmed experimentally that the autoregressive model outperforms the non-autoregressive model. Furthermore, we demonstrate that our model outperforms standard designs of conditional flows on classical image-to-image translation tasks such as image super-resolution, image colorization and image inpainting. In particular, we show that on these tasks CAFLOW outperforms former designs of conditional flows and GAN-based models thanks to its expressive autoregressive structure.

Concurrently to our work, Liang et al. [120] introduced autoregressive conditioning in the latent space, which yielded improved results compared to their earlier work [127]. However, they make the following restrictive modelling assumption: the low resolution image is part of the target latent space. This makes their method not applicable to general inverse problems in vision.

As far as inverse problems on other modalities are considered, our method can be suitably modified to address inverse problems on structured data where there is an underlying hierarchical decomposition of the signal. It is true that our method would not be easily applied in unstructured data such as categorical or tabular data. However, the same applies to all other conditional normalizing flow methods that use a multiscale structure.

Organization of the paper. The paper is organized as follows. In Section 3.2, we provide the necessary theoretical background on unconditional and conditional normalizing flows. In Section 3.3, we describe our CAFLOW method discussing the modeling assumptions and deriving a formula for estimating the log-likelihood of the unknown conditional probability. In Section 3.4, we present the quantitative and qualitative results of our experiments. Finally, in Sections 3.5 we discuss the limitations of our work and we draw our conclusions.

3.2 Background

3.2.1 Notations

Random variables will be denoted by capital letters, e.g. X, Y, while their associated probability will be indicated by $P(X)$, $P(Y)$ and their distributions by p_X and p_Y. We use calligraphic capital letters, e.g. $\mathscr{X}, \mathscr{Y}$, to denote the sample space of the random variables X, Y and small bold letters x, y for the samples. Finally, given two random variables X and Y we will consider the conditional probability of Y given X and we denote it by $P(Y|X)$. Its density will be indicated by $p_{Y|X}$. We remark that throughout the paper we will always implicitly assume that the considered probabilities admit a density.

3.2.2 Normalizing flows

Given a random variable Y defined on $\mathscr{Y}$ with an unknown distribution p_Y, the main idea of flow-based generative modeling is to approximate p_Y with a learned function p_Y^θ, which is parametrized by an invertible neural network G^θ mapping from a latent space $\mathscr{Z}$ to $\mathscr{Y}$. Choosing $P(Z)$ to be a probability associated to the random variable Z and p_Z its distribution on $\mathscr{Z}$, we define the function p_Y^θ as the distribution of the push-forward measure of P_Z by the map G^θ. By the change-of-variables formula for probability distributions, p_Y^θ is easily computed as

$$p_Y^\theta(y) = p_Z(F^\theta(y)) \left| \det \left(\frac{dF^\theta(y)}{dy} \right) \right|,$$

where F^θ is the inverse of G^θ. Thanks to the formula above it is then possible to match p_Y^θ to p_Y by miminizing the negative log-likelihood of p_Y^θ, defined as

$$-\log p_Y^\theta(y) = -\log(p_Z(F^\theta(y))) - \log \left| \det \left(\frac{dF^\theta(y)}{dy} \right) \right|.$$

In practice, the transformation G^θ is a composition of learnable invertible transformations $G_1, ..., G_n$ such that $G_i : \mathscr{Z}_i \to \mathscr{Z}_{i+1}$ (where $\mathscr{Z}_1 = \mathscr{Z}$ and $\mathscr{Z}_{n+1} = \mathscr{Y}$) for intermediate latent spaces $\mathscr{Z}_i$ and

$$y = G_n(G_{n-1}(...G_1(z))) = G^\theta(z), \quad z = F_1(F_2(...F_n(y))) = F^\theta(y), \tag{3.1}$$

where $F_i = (G_i)^{-1}$ for every i and thus $F^\theta = (G^\theta)^{-1}$ (Note that we are assuming implicitly that each G_i and F_i depends on θ, but we drop the superscript θ for notational convenience).

This model is called a normalizing flow. In this case, using the properties of the Jacobian for composition of invertible transformations the log-likelihood of p_Y^θ can be computed as

$$\log(p_Y^\theta(y)) = \log(p_Z(F^\theta(y))) + \sum_{i=1}^{n} \log \left| \det \left(\frac{dF_i(z_{i+1})}{dz_{i+1}} \right) \right|, \tag{3.2}$$

where $z_{n+1} = y$.

In order to estimate $\log(p_Y^\theta(y))$ the transformations G_i need to be designed to have a computable inverse F_i and a tractable Jacobian determinant. We next present typical choices for the transformations G_i. We describe each method by presenting the forward transform G_i, the inverse transform F_i and the log-determinant of the Jacobian. The calculation of the Jacobian determinant is used in the calculation of the log-likelihood in (3.2).

Affine coupling layer. Affine coupling layers [46] capture complex dependencies of the activations in an invertible and computationally efficient way.

Forward Function:

A given a tensor z is partitioned into two parts: z_1 and z_2. The tensor z_2 remains unchanged and is used in the transformation of z_1. A nonlinear mapping, represented by NN (typically a shallow convolutional neural network), processes z_2 producing two outputs of the same dimension as z_1: u and t. The exponential of u is used as the scale and t as the bias of the affine transformation of z_1. The steps are the following:

$$\begin{aligned} z_1, z_2 &= \text{split}(z) \\ (u,t) &= NN(z_2) \\ s &= \exp(u) \\ y_1 &= s \odot z_1 + t \\ y_2 &= z_2. \end{aligned}$$

The final tensors y_1 and y_2 merge to produce the output tensor y.

Inverse Function:

$$\begin{aligned} y_1, y_2 &= \text{split}(y) \\ (u,\mathbf{t}) &= NN(y_2) \\ s &= \exp(u) \\ z_1 &= \frac{(y_1 - t)}{s} \\ z_2 &= y_2. \end{aligned}$$

Both z_1 and z_2 merge to produce the reversed tensor z.

Log-Determinant of the Jacobian: The log-determinant of the Jacobian can be computed as:

$$\sum_{i=1}^{d} (\log |s_i|)$$

where the sum is over all the elements of the tensor s.

Invertible 1×1 **Convolution**. In the Affine coupling layer, the partitioning of the tensor **z** is static, implying that stacking multiple Affine coupling layers still leaves part of the tensor unchanged. This limits the range of inverse functions the model can represent. Addressing this limitation, Kingma et al. [100] introduced the invertible 1×1 convolution layer. By blending information across the channel dimension prior to the Affine coupling layer, the invertible 1×1 enhances the model's capacity to represent inverse functions of increased complexity.

Forward Function:

$$\forall i, j : y_{i,j} = W z_{i,j},$$

where:

- W is a weight matrix of size $c \times c$
- $z_{i,j}$ and $y_{i,j}$ are spatial indices into tensors z and y, respectively, both of size $h \times w \times c$.

Inverse Function:

$$\forall i, j : z_{i,j} = W^{-1} y_{i,j}.$$

Log-Determinant of the Jacobian:

$$h \cdot w \cdot \log |\det(W)|$$

The computational cost of computing the log-determinant of the Jacobian can be improved from $O(c^3)$ to $O(c)$ if W is parametrized in its LU decomposition as follows:

$$W = PL(U + \text{diag}(\mathbf{r})),$$

where

- P is a permutation matrix that remains fixed,
- L is a lower triangular matrix with ones on the diagonal,
- U is an upper triangular matrix with zeros on the diagonal,

- **r** is a vector.

Here, P, L, U and **r** are initialized by sampling a random rotation matrix W and then computing the values of P, L, U and **r** that correspond to the LU decomposition of W. The matrix P remains fixed, while L, U and **r** are learnable. By parametrizing W in its LU decomposition, the computational complexity is reduced from $O(c^3)$ to $O(c)$, because $\log|\det(W)| = \sum_{i=1}^{c}(\log|r_i|)$, where $(r_1, ..., r_c)$ are the elements of the vector r.

Actnorm. Actnorm, presented in [12], is a modification of the conventional batch normalization [9]. It is tailored to handle stability challenges in training deep models, especially in scenarios where memory constraints mandate a very small batch size per processing unit. Rather than depending on batch statistics, Actnorm executes an affine transformation on activations using a learnable scale and bias parameter per channel. Each channel undergoes unique scaling and biasing. The initial values of the parameters depend on the data, ensuring that the post-actnorm activations per channel have zero mean and unit variance. After this initialization, these parameters are considered trainable and evolve independently of the input data.

Forward Function:

$$\forall i, j : y_{i,j} = s \odot z_{i,j} + b,$$

where:

- $z_{i,j}$ and $y_{i,j}$ are spatial indices into tensors z and y, both of which are of size $h \times w \times c$.
- s and b are scale and bias parameters, respectively.

Inverse Function:

$$\forall i, j : z_{i,j} = \frac{(y_{i,j} - b)}{s}.$$

Log-Determinant of the Jacobian: The log-determinant of the Jacobian can be computed as:

$$h \cdot w \cdot \sum_{i=1}^{d}(\log|s_i|).$$

These layers are the building blocks of normalizing flows. A typical normalizing flow architecture consists of sequential blocks where each block contains Actnorm layer, an invertible 1x1 convolution layer, and an affine coupling layer in that order. More details about how these layers are utilised to form the normalizing flow architecture are provided in A.1.1.

3.2.3 Conditional Normalizing Flows

The normalizing flow approach can be adapted to model conditional densities of complicated target distributions. Precisely, given two random variables X and Y, the conditional distribution $p_{Y|X}$ is parametrized using a transformation $G^\theta : \mathscr{Z} \times \mathscr{X} \to \mathscr{Y}$ such that $G^\theta(\cdot,x) : \mathscr{Z} \to \mathscr{Y}$ is invertible for every condition x. Choosing $P(Z)$ to be a probability associated to the random variable Z defined on $\mathscr{Z}$ and p_Z its distribution on $\mathscr{Z}$, we denote by $p^\theta_{Y|X}$ the distribution of the probability obtained by applying the push-forward by the map $z \mapsto G^\theta(z,x)$ to $P(Z)$. By the change-of-variables formula for probability distributions the conditional log-likelihood of $p^\theta_{Y|X}$ can be then computed as

$$\log(p^\theta_{Y|X}(y|x)) = \log(p_Z(F^\theta(y,x))) + \log\left|\det\left(\frac{\partial F^\theta(y,x)}{dy}\right)\right|, \tag{3.3}$$

where $F^\theta(\cdot,x)$ is the inverse of $G^\theta(\cdot,x)$ for every condition x. Consequently, a generative model for $p_{Y|X}$ can be trained by minimizing the negative log-likelihood of the parameters θ using the formula in (3.3). The sampling procedure works similarly to standard normalizing flows. We generate a sample $y \sim p^\theta_{Y|X}(\cdot,x)$ by sampling a latent $z \sim p_Z$ and passing it through $G^\theta(\cdot,x)$, yielding $y = G^\theta(z,x)$. Similarly to traditional normalizing flows the map G^θ is modeled through a composition of learnable invertible conditional transformations. We describe next the specific layers used in conditional normalizing flow architectures.

Conditional affine coupling layer. As a variant of the affine coupling layer, the conditional affine coupling layer integrates an external condition, allowing for conditional modelling. The scale and bias are computed via parametrized functions that consider not only the split activation z_2, but also the external condition x.

Forward Function: A given tensor z is equally divided into two fixed parts: z_1 and z_2.

Similarly to the unconditional case, a nonlinear mapping represented by *NN* processes z_2 and the external condition x producing two outputs of the same dimension as z_1: u and t. The exponential of u is used as the scale and t as the bias of the affine transformation of z_1. The steps are the following:

$$
\begin{aligned}
z_1, z_2 &= \text{split}(z) \\
(u,t) &= NN(z_2, x) \\
s &= \exp(u) \\
y_1 &= \mathbf{s} \odot z_1 + t \\
y_2 &= z_2.
\end{aligned}
$$

The resulting tensors, y_1 and y_2, are combined to generate the output tensor y.

Inverse Function: Reversing the forward function and accounting for the condition x, we get:

$$
\begin{aligned}
y_1, y_2 &= \text{split}(y) \\
(u,t) &= NN(y_2, x) \\
s &= \exp(u) \\
z_1 &= \frac{(y_1 - t)}{s} \\
z_2 &= y_2.
\end{aligned}
$$

Both z_1 and z_2 merge to yield the original tensor z.

Log-Determinant of the Jacobian: The log-determinant of the Jacobian can be computed as:

$$\sum_{i=1}^{d} (\log |s_i|).$$

Affine Injector. The Affine Injector layer, introduced in [127], enhances the capability of normalizing flows in transferring information from the conditioning data. Unlike the conditional Affine coupling layer, which influences only a subset of the intermediate activation z, the Affine Injector impacts all channels and spatial locations of the activation, thereby enabling more comprehensive data flow from the condition to the main branch of the flow. Here, the "main branch of the flow" denotes the series of transformations that process the data tensor and its latent intermediate activations. The scale and bias tensors are parametrized functions of only the condition x.

Forward Function: Given the external condition x, the nonlinear mapping represented by NN processes x to produce two outputs of the same dimension as z: u and t. The exponential of u is used as the scale and t as the bias for the affine transformation of z. The steps are the following:

$$
\begin{aligned}
(u,t) &= NN(x) \\
s &= \exp(u) \\
y &= s \odot z + t.
\end{aligned}
$$

Inverse Function: Reversing the forward function and accounting for the condition x, we get:

$$
\begin{aligned}
(u,t) &= NN(x) \\
s &= \exp(u) \\
z &= \frac{y-t}{s}.
\end{aligned}
$$

Log-Determinant of the Jacobian: The log-determinant of the Jacobian can be computed as:

$$\sum_{i=1}^{d} (\log |s_i|).$$

As empirically demonstrated by Lugmayr et al. [127], the integration of the Affine Injector augments the modeling capacity of the conditional normalizing flow.

3.2.4 Conditioning in the Normalizing Flow latent space

An alternative way for conditional generation using normalizing flows is proposed in [193] in a model named Dual-Glow. The authors couple two multi-scale normalizing flows modeled with a GLOW architecture that interact at different scales. In particular, denoting by Y and W[1] the random variables modeling the distribution of conditioning images and of conditioned images (respectively), they estimate the conditional probability $P(W|Y)$ using two multi-scale normalizing flows which convert images Y and W into sequences of hierarchical latent random variables $[D_{n-1}, ..., D_0]$ and $[L_{n-1}, ..., L_0]$, respectively, as shown in schematically in Figure 3.1. Then, they model the distribution of the latent random variables $[L_{n-1}, ..., L_0]$ given the latent random variables $[D_{n-1}, ..., D_0]$ by making the assumption that information

[1] In this section, W denotes a random variable, distinct from its previous use as a weight matrix. The notation for the weight matrix will not be used further in this article.

is exchanged only between latent spaces of the same scale, that is

$$P(L_{n-1},...,L_0|D_{n-1},...,D_0) = \prod_{i=0}^{n-1} P(L_i|D_i). \tag{3.4}$$

The final step is the modelling of the each conditional distribution $P(L_i|D_i)$. They make the assumption that the conditional probability $P(L_i|D_i)$ is a multidimensional Gaussian distribution, whose mean μ_i^θ and covariance Σ_i^θ are parametrised functions of D_i. In particular, its density is parametrized by the functions $p^\theta_{L_i|D_i}$ given by

$$p^\theta_{L_i|D_i}(\ell_i, d_i) = \mathcal{N}(\ell_i; \mu_i^\theta(d_i), \Sigma_i^\theta(d_i)) \quad i = 0,\dots,n-1. \tag{3.5}$$

The decomposition of conditioning and conditioned images into sequences of hierarchical latent spaces that exchange information at different scales is a natural approach to conditional probability estimation. Indeed, it offers the possibility to prescribe the rate of information exchange between different scales and to monitor the information flow in a more accurate way. In the next section we will discuss how our method is taking advantage of such flexibility, by decomposing the conditional distribution of the latent encodings and decodings in its autoregressive components.

3.3 Method

The aim of our CAFLOW model is to perform image-to-image translation tasks by learning the conditional distribution $P(W|Y)$ where W and Y are random variables that model given image distributions. Following a similar architecture to [193] we use two multi-scale normalizing flows R^θ and T^θ to convert images Y and W into two sequences of n hierarchical latent variables defined on latent space of decreasing dimension: $R^\theta(Y) = \tilde{D}_n$, where $\tilde{D}_n := [D_{n-1},...,D_0]$ and $T^\theta(W) = \tilde{L}_n$, where $\tilde{L}_n := [L_{n-1},...,L_0]$. Then we design an autoregressive model based on conditional normalizing flows to learn the conditional distribution $P(\tilde{L}_n|\tilde{D}_n)$.

3.3.1 Modeling assumptions

The conditional distribution $P(\tilde{L}_n|\tilde{D}_n)$ is factorized using the chain rule of probability into n autoregressive component distributions as shown in (3.6):

$$P(\tilde{L}_n|\tilde{D}_n) = \prod_{i=0}^{n-1} P(L_i|\tilde{L}_i, D_{n-1},...,D_0) \tag{3.6}$$

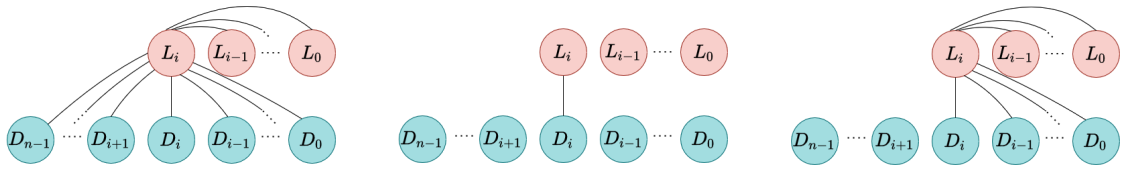


Fig. 3.1 From left to right: ideal dependencies in the i^{th} autoregressive component. Dual-Glow modeling assumption [193]; information is exchanged only between latent spaces having the same dimension. Our modeling assumption; we retain the dependencies between L_i and the latent spaces of lower dimension.

with the notational convention that $\tilde{L}_0 = \emptyset$ and $\tilde{L}_i = [L_{i-1}, \dots, L_0]$ for every i. The dependencies of the i^{th} component distribution are shown graphically in the left diagram of Figure 3.1. Modeling those n autoregressive distributions can be unnecessarily computationally expensive. For this reason, we assume that

$$P(\tilde{L}_n|\tilde{D}_n) = \prod_{i=0}^{n-1} P(L_i|\tilde{L}_i, \tilde{D}_{i+1}), \tag{3.7}$$

where $\tilde{D}_{i+1} = [D_i, \dots, D_0]$. In particular, we retain the dependencies between L_i and all L and D latent variables of level i and below, which effectively means that we are pruning only the dependencies between L_i and $D_{i+1}, ..., D_{n-1}$. We advocate that this is a valid assumption, because the encoded information in the split variables of the multi-scale flow typically ranges from local noise patterns and image details to higher-level information as we move from the early split variables to the final split variables. The hierarchical representation in multi-scale normalizing flows is attributed to the interplay of depth of computation and the information bottleneck. We refer the reader to Appendix A.2 for a more detailed discussion.

We remark that our modeling assumption is weaker than the one implicitly used in [193] described in (3.5) (see Figure 3.1). Precisely we allow information to be exchanged between latent spaces of different dimension. As we will demonstrate in the experiments our choice allows for more expressive architectures able to capture correlations between different scales. We represent schematically in Figure 3.1 the difference between the theoretical latent space dependencies, the one assumed in [193] and our modeling choice. Under this modeling assumption, our goal is to estimate the conditional distributions $P(L_i|\tilde{L}_i, \tilde{D}_{i+1})$ for every i and then recover $P(\tilde{L}_n|\tilde{D}_n)$ using (3.7).

3.3.2 Modeling the autoregressive components using conditional normalizing flows

Modeling the autoregressive components $P(L_i|\tilde{L}_i, \tilde{D}_{i+1})$ in a computationally efficient way is challenging because of the dependence on the multiple latent factors of different dimension that encode different semantic content. We propose to estimate each autoregressive component $P(L_i|\tilde{L}_i, \tilde{D}_{i+1})$ with a novel multiscale normalizing flow architecture that efficiently incorporates the latent conditions. For every $i = 0, \dots, n-1$ we define a sequence of latent variables $\tilde{Z}_{i+1} := [Z_i^i, ..., Z_0^i]$ defined on latent spaces of decreasing dimension $\tilde{\mathscr{Z}}_{i+1} := [\mathscr{Z}_i^i, ..., \mathscr{Z}_0^i]$ and a parametrized transformation

$$G_i^\theta : \tilde{\mathscr{Z}}_{i+1} \times \tilde{\mathscr{D}}_{i+1} \times \tilde{\mathscr{L}}_i \to \mathscr{L}_i.$$

The transformations G_i^θ are constructed by assembling multi-scale transformations $(g_j^i)_{j=0}^i$ (implicitly depending on the parameter θ) defined as follows:

$$\begin{aligned}
&g_0^i : \mathscr{Z}_0^i \times \mathscr{D}_0 \times \mathscr{L}_0 \to \mathscr{Z}_0'^i \\
&g_j^i : \mathscr{Z}_{j-1}'^i \times \mathscr{Z}_j^i \times \mathscr{D}_j \times \mathscr{L}_j \to \mathscr{Z}_j'^i \quad \text{for} \quad j = 1, ..., i-1 \\
&g_i^i : \mathscr{Z}_{i-1}'^i \times \mathscr{Z}_i^i \times \mathscr{D}_i \to \mathscr{L}_i
\end{aligned}$$

where $\mathscr{Z}_{i-1}'^i, \dots, \mathscr{Z}_0'^i$ are intermediate latent spaces of decreasing dimension, g_j^i is invertible as a function from $\mathscr{Z}_{j-1}'^i \times \mathscr{Z}_j^i$ to $\mathscr{Z}_j'^i$ and g_0^i is invertible as function from $\mathscr{Z}_0^i$ to $\mathscr{Z}_0'^i$. The transformation G_i^θ is then obtained by composing the functions g_j^i in the following way. Given the conditioning variables $(d_j)_{j=0}^i \in \tilde{\mathscr{D}}_{i+1}$ and $(l_j)_{j=0}^{i-1} \in \tilde{\mathscr{L}}_i$, a latent variable $z_0 \in \mathscr{Z}_0^i$ is transformed to $z_0' \in \mathscr{Z}_0'^i$ by function $g_0^i(\cdot; d_0, l_0)$. Then z_0' and z_1 are concatenated and inserted to $g_1^i(\cdot; d_1, l_1)$ which outputs $z_1' \in \mathscr{Z}_1'^i$. This process continues as implied up until the i^{th} level whose output is $l_i = g_i^i(z_{i-1}', z_i; d_i) \in \mathscr{L}_i$. A schematic description of G_i^θ is presented in the right diagram of Figure 3.2.

Denote by f_j^i (implicitly depending on the parameter θ) the inverse of g_j^i for fixed conditioning variables in $\mathscr{D}_j$ and $\mathscr{L}_j$

$$\begin{aligned}
&f_0^i : \mathscr{Z}_0'^i \times \mathscr{D}_0 \times \mathscr{L}_0 \to \mathscr{Z}_0^i \\
&f_j^i : \mathscr{Z}_j'^i \times \mathscr{D}_j \times \mathscr{L}_j \to \mathscr{Z}_{j-1}'^i \times \mathscr{Z}_j^i \quad \text{for} \quad j = 1, ..., i-1 \\
&f_i^i : \mathscr{L}_i \times \mathscr{D}_i \to \mathscr{Z}_{i-1}'^i \times \mathscr{Z}_i^i
\end{aligned}$$

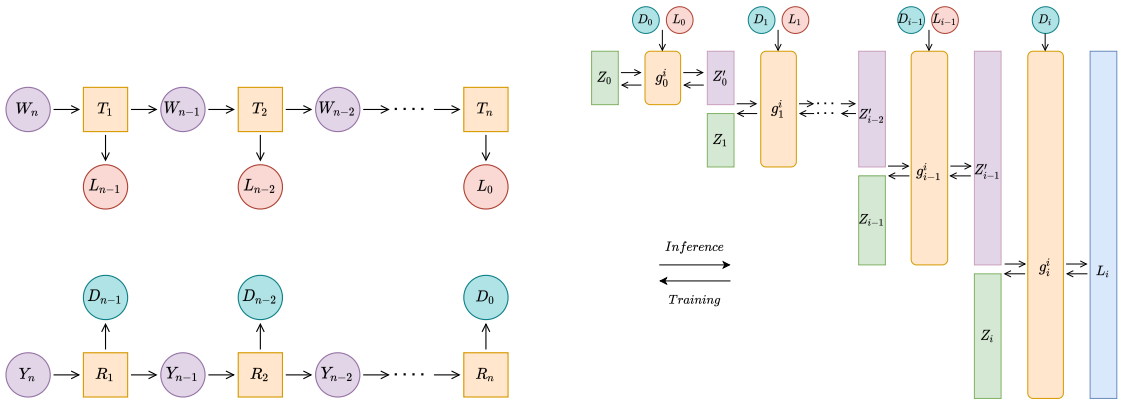


Fig. 3.2 Left: unconditional normalizing flow architecture used to encode conditioning and conditioned images, denoted by $Y_n = Y$ and $W_n = W$ respectively, into a sequence of hierarchical latent variables. Right: design of the conditional transformation G_i^θ that models the i^{th} autoregressive component. The index of the flow i is omitted in both the transformed latent variable Z_j and the intermediate latent variables Z'_j for simplicity.

and by F_i^θ the inverse of G_i^θ as a function from $\tilde{\mathscr{Z}}_{i+1}$ to $\mathscr{L}_i$ obtained by composing the functions f_j^i for $j = 0, \ldots, i$. We model each f_j^i by adopting the conditional flow design of [127]. More specifically, we initially use the same squeeze layer, which is followed by two transition steps and K conditional flow steps. Each transition step consists of an actnorm layer followed by 1×1 invertible convolution layer. Each conditional flow step consists of an actnorm layer followed by 1×1 invertible convolution, which is followed by an affine injector and an affine coupling layer. We use from 8 to 16 conditional flow steps depending on the difficulty of the image translation task.

A discussion on sharing weights strategies. It is worth noting that the multi-scale transformations, denoted as g_j^i, for all transformations G_i^θ where $i \geq j+1$, share the same architecture and function – integrating information from conditions D_j and L_j. This hints at the potential of utilizing a singular subflow, g_j^{j+1}, for every G_i^θ with $i \geq j+1$. Such a strategy could decrease the computational burden and introduce an inductive bias. Yet, it is crucial to recognize that this approach assumes the only distinctive subflow in each G_i^θ flow is g_i^i. All the deeper subflows, represented by g_k^i where $k < i$, are shared between G_i^θ and other G_j^θ flows. This results in a modeling constraint, narrowing the space of functions the architecture can represent. Our preliminary tests confirmed this limitation; the performance using this approach significantly lags behind the more generalized model presented in this paper.

3.3.3 Maximum log-likelihood estimation and training

Here we use the constructed transformations $G_i^\theta : \tilde{\mathscr{Z}}_{i+1} \times \tilde{\mathscr{D}}_{i+1} \times \tilde{\mathscr{L}}_i \to \mathscr{L}_i$ to parametrize each autoregressive component $P(L_i|\tilde{L}_i, \tilde{D}_{i+1})$ and, together with the unconditional normalizing flows $R^\theta := R_n^\theta \circ \ldots \circ R_1^\theta$ and $T^\theta := T_n^\theta \circ \ldots \circ T_1^\theta$, use them to estimate $P(W|Y)$. To model $P(\tilde{L}_n|\tilde{D}_n)$ through its autoregressive componenents as in (3.6), we define the operator $G^\theta : \tilde{\mathscr{Z}}_1 \times \ldots \times \tilde{\mathscr{Z}}_n \times \tilde{\mathscr{D}}_n \to \tilde{\mathscr{L}}_n$ as the concatenation of the operators G_i^θ defined in the previous section, i.e.

$$G^\theta = [G_0^\theta, \ldots, G_{n-1}^\theta]. \tag{3.8}$$

where G_i^θ is computed inductively using G_j^θ for $j < i$ as in the autoregressive factorization (3.6). Then we consider the map $H^\theta : \tilde{\mathscr{Z}}_1 \times \ldots \times \tilde{\mathscr{Z}}_n \times \mathscr{Y} \to \mathscr{W}$ defined as $H^\theta(\tilde{z}_1, \ldots, \tilde{z}_n, y) := (T^\theta)^{-1}(G^\theta(\tilde{z}_1, \ldots, \tilde{z}_n; R^\theta(y)))$ and we denote by $p_{W|Y}^\theta$ the distribution of the pushforward measure

$$H_\#^\theta(\cdot, y)P([\tilde{Z}_1, \ldots, \tilde{Z}_n]). \tag{3.9}$$

We minimize the negative log-likelihood of $p_{W|Y}^\theta$ to obtain an estimate of $P(W|Y)$. With this aim, for every latent variable Z_j^i in $\tilde{Z}_j$ we choose as prior a multivariate normal distribution, whose density we denote by $\mathscr{N}(\cdot; \mathbf{0}, \mathbf{I})$. In the next theorem we provide an explicit formula for the logarithm of $p_{W|Y}^\theta$ expressed in terms of R_i^θ, T_i^θ and f_j^i.

Proposition 3.3.1. *The logarithm of $p_{W|Y}^\theta$ can be computed as*

$$\begin{aligned}
\log p_{W|Y}^\theta(w|y) = &\sum_{i=1}^{n} \log \left| \det \frac{\partial T_i^\theta(w_{n-i+1})}{\partial w_{n-i+1}} \right| \\
&+ \sum_{i=1}^{n-1} \sum_{j=0}^{i-1} \left[\log \mathscr{N}(z_j^i; \boldsymbol{0}, \boldsymbol{I}) + \log \left| \det \left(\frac{\partial f_j^i(z_j'^i; l_j, d_j)}{\partial z_j'^i} \right) \right| \right] \\
&+ \sum_{i=0}^{n-1} \left[\log \mathscr{N}(z_i^i; \boldsymbol{0}, \boldsymbol{I}) + \log \left| \det \left(\frac{\partial f_i^i(l_i; d_i)}{\partial l_i} \right) \right| \right]
\end{aligned} \tag{3.10}$$

where the normal distributions are computed in the latent variables $z_j^i = f_j^i(z_j'^i, d_j, l_j)$ and $z_i^i = f_i^i(d_i, l_i)$. Moreover, each intermediate latent variable $z_j'^i$ is defined recursively by the application of the previous f_j^i (see Figure 3.2) and $d_i \in \mathscr{D}_i$ implicitly depend on $y \in Y$ through the normalizing flow R^θ.

Proof. First we express the logarithm of $p^{\theta}_{W|Y}$ in terms of $p^{\theta}_{\tilde{L}_n|\tilde{D}_n}$, where $p^{\theta}_{\tilde{L}_n|\tilde{D}_n}$ is the distribution induced by the map G^{θ} defined in (3.8). We obtain

$$\begin{aligned}
\log p^{\theta}_{W|Y}(w|y) &= \log\left|\det\left(\frac{\partial T^{\theta}(w)}{\partial w}\right)\right| + \log p^{\theta}_{\tilde{L}_n|\tilde{D}_n}(\tilde{l}_n|\widetilde{d}_n) \\
&= \sum_{i=1}^{n}\log\left|\det\frac{\partial T^{\theta}_i(w_{n-i+1})}{\partial w_{n-i+1}}\right| + \log\prod_{i=0}^{n-1} p^{\theta}_{L_i|\tilde{L}_i,\tilde{D}_{i+1}}(l_i|\tilde{l}_i,\widetilde{d}_{i+1}) \qquad (3.11)\\
&= \sum_{i=1}^{n}\log\left|\det\frac{\partial T^{\theta}_i(w_{n-i+1})}{\partial w_{n-i+1}}\right| + \sum_{i=0}^{n-1}\log p^{\theta}_{L_i|\tilde{L}_i,\tilde{D}_{i+1}}(l_i|\tilde{l}_i,\widetilde{d}_{i+1}).
\end{aligned}$$

The justification for the first line can be found in [193, Section 3]. In the second line we use the chain rule for distribution and we factorize the conditional distribution using the dependency assumptions of our model. Now observe that

$$\begin{aligned}
\sum_{i=0}^{n-1}\log p^{\theta}_{L_i|\tilde{L}_i,\tilde{D}_{i+1}}(l_i|\tilde{l}_i,\widetilde{d}_{i+1}) &= \sum_{i=0}^{n-1}\log p_{\tilde{Z}_{i+1}}(F^{\theta}_i(l_i;\tilde{l}_i,\widetilde{d}_{i+1})) \\
&\quad + \log\left|\det\frac{\partial F^{\theta}_i(l_i;\tilde{l}_i,\widetilde{d}_{i+1})}{\partial l_i}\right| \\
&= \sum_{i=1}^{n-1}\sum_{j=0}^{i-1}\left[\log\mathcal{N}(z^i_j;\mathbf{0},\mathbf{I})\right. \qquad (3.12)\\
&\quad \left. + \log\left|\det\left(\frac{\partial f^i_j(z'^i_j;l_j,d_j)}{\partial z'^i_j}\right)\right|\right] \\
&\quad + \sum_{i=0}^{n-1}\left[\log\mathcal{N}(z^i_i;\mathbf{0},\mathbf{I}) + \log\left|\det\left(\frac{\partial f^i_i(l_i;d_i)}{\partial l_i}\right)\right|\right],
\end{aligned}$$

where the normal distributions are computed in the latent variables $z^i_j = f^i_j(z'^i_j, d_j, l_j)$ and $z^i_i = f^i_i(d_i, l_i)$. The first line of (3.12) is obtained by using the change-of-variables formula. In the second line we use the chain rule for the composition of functions using the definition of F^{θ}_i in 3.3.2 and the assumption that the latent variables $Z^i_i, Z^i_{i-1}, ..., Z^i_0$ which comprise $\tilde{Z}_{i+1}$ are i.i.d. with distribution $\mathcal{N}(\mathbf{0},\mathbf{I})$. The sum is then broken into two sums, because the f^i_i functions are conditioned only on d_i by definition. Therefore, combining (3.11) and (3.12) we obtain

$$\begin{aligned}\log p^{\theta}_{W|Y}(w|y) = &\sum_{i=1}^{n} \log \left| \det \frac{\partial T_i^{\theta}(w_{n-i+1})}{\partial w_{n-i+1}} \right| \\ &+ \sum_{i=1}^{n-1} \sum_{j=0}^{i-1} \left[\log \mathcal{N}(z^i_j; \mathbf{0}, \mathbf{I}) + \log \left| \det \left(\frac{\partial f^i_j(z'^i_j; l_j, d_j)}{\partial z'^i_j} \right) \right| \right] \\ &+ \sum_{i=0}^{n-1} \left[\log \mathcal{N}(z^i_i; \mathbf{0}, \mathbf{I}) + \log \left| \det \left(\frac{\partial f^i_i(l_i; d_i)}{\partial l_i} \right) \right| \right]\end{aligned}$$

as we wanted to prove.

□

We train our CAFLOW model by minimizing the following training objective

$$\log p^{\theta}_{W|Y}(w|y) + \lambda \log p^{\theta}_{Y}(y),$$

where $p^{\theta}_{Y}(y)$ is estimated using the unconditional normalizing flow R^{θ} as in Subsection 3.2.2. The parameter λ interpolates between the conditional distribution $P(W|Y)$ (when $\lambda = 0$) and the joint distribution $P(Y,W)$ (when $\lambda = 1$). Indeed, if $\lambda = 1$ the objective can be written as

$$\log p^{\theta}_{W|Y}(w|y) + \log p^{\theta}_{Y}(y) = \log(p^{\theta}_{W|Y}(w|y) p^{\theta}_{Y}(y)) = \log \left(p^{\theta}_{Y,W}(y,w) \right) \tag{3.13}$$

where $p^{\theta}_{Y,W}$ is the density of the joint distribution $P(Y,W)$. Note that λ can be interpreted as a regularization parameter. This intuition is justified noticing that if $\lambda = 1 - \varepsilon$ for $\varepsilon \in [0,1]$ we can write

$$\log p^{\theta}_{W|Y}(w|y) + (1-\varepsilon) \log p^{\theta}_{Y}(y) = \log(p^{\theta}_{Y,W}(y,w)) + \log \left(\frac{1}{(p^{\theta}_{Y}(y))^{\varepsilon}} \right) \tag{3.14}$$

where the term $\log(\frac{1}{(p^{\theta}_{Y}(y))^{\varepsilon}})$ is close to infinity in the regions where $p^{\theta}_{Y}(y)$ is small, that are the unlikely images in the distribution. By choosing $\varepsilon < 1$ instead of $\varepsilon \sim 1$ the term $\log(\frac{1}{(p^{\theta}_{Y}(y))^{\varepsilon}})$ becomes smaller in such regions by improving the stability of the training process.

3.3.4 Inference

The standard way to perform inference using a trained CAFLOW model is the following:

1. Calculate the conditional encodings $(d_{n-1},...,d_0) \in \tilde{\mathscr{D}}_n$ by passing the conditioning image through the unconditional multi-scale flow R^θ.
2. Sample latent variables z_i^j from $\mathscr{N}(\mathbf{0}, \tau^2\mathbf{I})$, where $0 < \tau \leq 1$ denotes the sampling temperature.
3. Calculate the output image latent variables $l_0,...,l_{n-1} \in \tilde{\mathscr{L}}_n$ by applying the transformations G_i^θ sequentially from G_0^θ to G_{n-1}^θ.
4. Finally, convert the output image latents $l_0,...,l_{n-1}$ to the output image by passing them through the unconditional reverse normalizing flow $(T^\theta)^{-1}$.

Sampling temperature τ and its importance. During training, we set $\tau = 1$. However, for sampling, we adopt a reduced temperature $\tau < 1$.

Despite sampling from a slightly different distribution than the one that we learnt, this choice enhances the quality of the output image reducing the occurrence of severe artifacts.

The challenge with many normalizing flow architectures, including ours, lies in their structure: composing numerous analytically bi-Lipschitz functions increases the Lipschitz coefficient of the inverse flow, as detailed by [15] and [204]. The amplification of the Lipschitz constant compromises the numerical stability of the inverse flow which is used for sampling. Empirically, a lower sampling temperature ensures better numerical stability, which implies that the Lipschitz constant of the inverse flow constrained in the higher density regions is significantly smaller.

The choice of τ for sampling is crucial in this regard. For $\tau < 1$ we gain a better numerical stability while we could lose accuracy as a consequence of sampling from a different distribution. This trade-off can be quantified (for instance) in terms of the 2-Wasserstein distance between the distributions generated by the normalizing flow G for different levels of τ as follows:

$$\begin{aligned} W_2(G_\# \mathscr{N}(0,\mathbf{I}), G_\# \mathscr{N}(0,\tau\mathbf{I})) &\leq \mathrm{Lip}(G) W_2(\mathscr{N}(0,\mathbf{I}), \mathscr{N}(0,\tau\mathbf{I})) \\ &\leq \mathrm{Lip}(G)\sqrt{d(1+\tau-2\sqrt{\tau})} \end{aligned}$$

where $\mathrm{Lip}(G)$ is the Lipschitz constant of G, $G_\#\mu$ is the push-forward of the probability measure μ by G and d is the dimension of the normal distributions. In particular, since the sampling error is amplified by the Lipschitz constant of the normalizing flow, the interplay between accuracy and numerical stability that is regulated by the choice τ is complex and heavily depends on the data distribution and on the learnt normalizing flow.

Finally, we mention that alternative methods based either on the regularization of the training objective [140] or on a priori controls of the Lipschitz constant of the flow (and its

inverse) [204] might achieve similar outcomes to adjusting the sampling temperature. Such approaches, however, come with additional computational challenges and their analysis fall beyond the scope of this study.

Optimized sample selection using conditional likelihood estimation. Our framework can act both as conditional generator and as a conditional likelihood estimator. Figure 3.3 shows ten super-resolved versions of the low resolution image in decreasing order of conditional log-likelihood. We leverage the conditional likelihood estimation to automatically select the best generated samples and disregard bad samples.

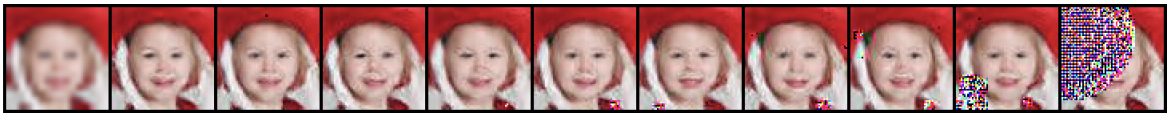


Fig. 3.3 10 super-resolved versions of the LR image in decreasing conditional log-likelihood order.

We have observed that CAFLOW typically assigns to the ground-truth image higher conditional log-likelihood than any generated sample. Such observation suggests that CAFLOW is a powerful conditional likelihood estimator, and that better samples can be obtained by the use of an optimisation algorithm, which searches for samples with high conditional likelihood. However, we used a simpler inference method in this work: keep the best N out of M generated samples based on the conditional log-likelihood. We empirically found that M should be larger for higher sampling temperatures.

3.4 Experiments

In this section we evaluate CAFLOW on three image-to-image translation tasks: image super-resolution, image colorization and image inpainting. Moreover, as ablation for the autoregressive structure in a image super-resolution task, we compare CAFLOW to a model that is conditioned in the normalizing flow latent space, assuming that information is exchanged only between latent spaces having the same dimension, c.f. Section 3.3.1 and Figure 3.1.

We refer the reader to section A.3 in the Appendix for more details about the reported experiments and to section A.4 for extended visual results.

3.4.1 Image Super-resolution

Ablation of the autoregressive structure

The idea of conditioning in the normalizing flow latent space was first introduced in [193] to design a conditional likelihood estimator named Dual-Glow. The main advantage of CAFLOW compared to Dual-Glow is the autoregressive structure in the latent space that allows to accommodate information exchange at different scales. We confirmed it experimentally by pruning the autoregressive connections and modeling the Dual-Glow component distribution $p(L_i|D_i)$ with a conditional normalizing flow. We named this model Dual-Glow+ as it has the same modeling structure as Dual-Glow with the extra advantage of modeling the component distribution $p(L_i|D_i)$ with a conditional normalizing flow instead of a multidimensional Gaussian distribution as in [193]. Moreover, we increased the depth of the conditional flow that models $p(L_i|D_i)$, so that Dual-Glow+ and CAFLOW have approximately the same number of parameters. This is important to ensure that improved performance of CAFLOW is the result of the autoregressive modeling structure.

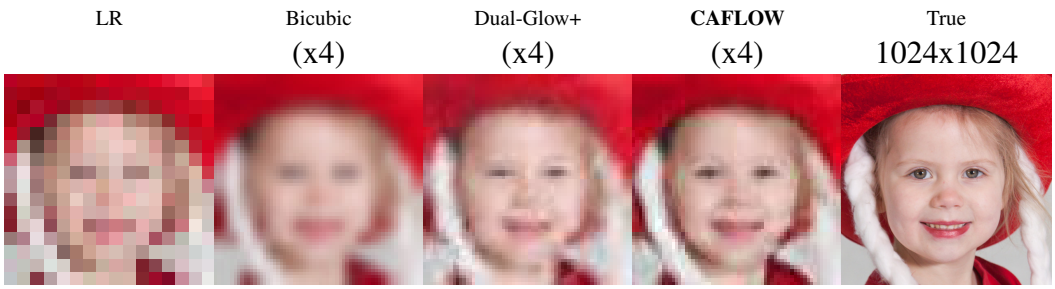


Fig. 3.4 Qualitative comparison of Dual-Glow+ and CAFLOW.

In Figure 3.4 we show the qualitative comparison of Dual-Glow+ and CAFLOW on a x4 super-resolution task. It is apparent that CAFLOW outperforms Dual-Glow+, confirming the significance of allowing exchange of information between different scales. We refer to the next section for more details about the image super-resolution experiment and a quantitative comparison between Dual-Glow+ and CAFLOW, c.f. Table 3.1.

Experiment

We evaluate the ability of CAFLOW and Dual-Glow+ to perform image super-resolution on the FFHQ dataset [94] (Creative Commons BY-NC-SA 4.0). We resized the original images to 16×16 and 64×64 patches and trained the model for x4 super-resolution. The quantitative evaluation is performed using the LPIPS and RMSE scores on 100 unseen

images. For inference, we used $\tau = 0.5$ and kept the sample with the highest conditional log-likelihood out of ten generated samples. A comparison with state-of-the-art methods is shown in Table 3.1. We present visual results in Figure 3.5 and refer the reader to the supplementary material for more examples.

Table 3.1 Quantitative evaluation of (x4) super-resolution on FFHQ 16^2. We report LPIPS/RMSE scores for each method. Lower scores are better.

Dataset	**CAFLOW**	**Dual-Glow+**	BRGM [129]	ESRGAN [208]	SRFBN [118]	BICUBIC
FFHQ 16^2	**0.08/17.56**	**0.14/18.56**	0.24/25.66	0.35/29.32	0.33/22.07	0.34/20.10

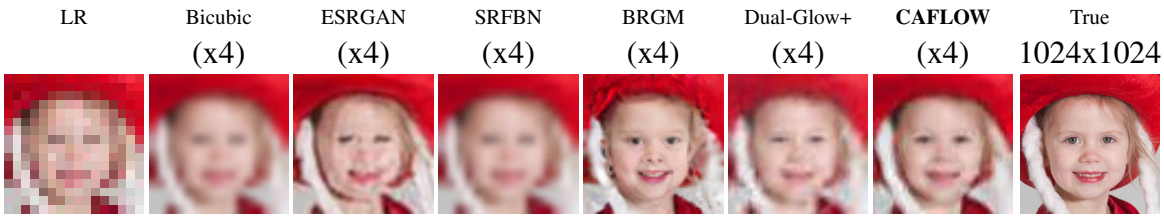


Fig. 3.5 Qualitative evaluation on FFHQ 4x super-resolution of 16x16 resolution images.

Table 3.1 shows that CAFLOW outperforms Dual-Glow+ based on both metrics, which confirms the theoretical advantage of the autoregressive modeling structure. Moreover CAFLOW outperforms all the other super-resolution methods based on both metrics. It is slightly inferior to BRGM [129] in terms of perceptual quality but it is significantly better in terms of fidelity which is reflected in the quantitative evaluation.

3.4.2 Image Colorization

To our knowledge, diverse image colorization has been addressed by conditional flows [4], conditional GANs [20] and recently score-based models [184]. We trained the model on 10% of the LSUN bedroom 64×64 training dataset [218], a popular dataset for image colorization. For inference, we used $\tau = 0.85$ and kept the best out of ten generated samples for each test image using the conditional log-likelihood. We report the performance of the model in Table 3.2. We use the FID score to compare our method against the cINN and ColorGAN, which have been trained on the full dataset. We show visual results for all methods in Figure 3.6. We did not include [184] in the comparison because it was trained on higher resolution images.

Our model outperforms both methods on image colorization based on the FID metric. We calculated the FID score by generating one conditional sample for each test image, so that

Fig. 3.6 Qualitative evaluation: Four colorizations proposed by CAFLOW, CINN and ColorGAN for three test images. ColorGAN generates unrealistically diverse colorizations with significant color artifacts (for example a yellow region on a white wall). CINN generates more realistic, less diverse colorizations with fewer pronounced color artifacts compared to ColorGAN, which is reflected in the improved FID score. Finally, CAFLOW generates even more realistic and less diverse colorizations than CINN with even rarer color artifacts, which is more representative of the data distribution according to the FID score.

Table 3.2 Quantitative evaluation of colorization on LSUN BEDROOM 64×64 dataset. We report FID score for each method. Lower scores are better.

Metric	**CAFLOW**	CINN [4]	ColorGAN [20]
FID	**18.15**	26.48	28.31

our evaluation is identical with the evaluation of the other methods. Moreover, we calculated the FID score by generating 5 samples for each test image, which yielded an improved FID score (**16.73**). We suggest that this protocol be adopted by methods which address diverse image colorization in the future.

3.4.3 Image Inpainting

We evaluated the performance of CAFLOW on image inpainting by removing central masks covering 25% of the centrally cropped human face of the CelebA dataset [122]. We compare the performance of the model with the conditional flow [125] on the same task using the PSNR metric, see Table 3.3. We show inpainting examples in Figure 3.7.

Our model outperforms [125] based on the PSNR metric and on qualititative performance. CAFLOW generates realistic images by blending smoothly the conditioning with the synthesized part of the image in contrast to [125] which generates overly smooth synthesized parts which do not blend well with the surrounding image. Both methods fail in faces which wear sunglasses or face sidewise. We believe that this is attributed to the small number of such training examples.

Fig. 3.7 Different inpaintings proposed by CAFLOW with $\tau = 0.5$. Ground truth on the right.

Table 3.3 Quantitative evaluation of inpainting on the CelebA dataset. We report PSNR and LPIPS scores for each method.

Method	PSNR↑	LPIPS↓
CAFLOW	**26.08**	0.06
[125]	24.88	-

3.5 Conclusion and Limitations

We have introduced a conditional autoregressive flow, coined CAFLOW, which combines autoregressive modeling with conditional normalizing flows for image-to-image translation. CAFLOW is an efficient conditional image generator and conditional likelihood estimator able to cross-correlate information at different scales, improving on the expressivity of former conditional flow architectures. We demonstrate its efficiency as conditional generator on standard image-to-image translation tasks such as image super-resolution, image colorization and image inpainting. We find that, in the above-mentioned tasks, CAFLOW achieves better performance than standard conditional flows and conditional GANs. In particular, we observe that the autoregressive modeling that allows information exchange at different scales, improves on Dual-Glow performances [193], where correlation is allowed only between latent spaces of the same dimension.

The main modeling limitation of the framework is attributed to limited expressivity of normalizing flows. Each transformation of a normalizing flow has to be invertible with a tractable calculation of the determinant of the Jacobian. This limits the representational power of normalizing flows. However, Wu et al. [213] claim to overcome expressivity limitations of normalizing flows by combining deterministic invertible functions with stochastic

sampling blocks. Therefore, incorporating their proposed stochastic blocks in our conditional autoregressive flows could potentially lead to significant performance improvement.

Acknowledgements

We thank Lynton Ardizzone for informing us about efficient ways to train normalizing flows. We also thank Razvan Marinescu and You Lu for providing help with practical details of the evaluation procedure.

GB acknowledges support from GSK. MC acknowledges support from the Royal Society (Newton International Fellowship NIF\R1\192048 Minimal partitions as a robustness boost for neural network classifiers). CE acknowledges support from the Wellcome Innovator Award RG98755. ZK acknowledges funding from the Royal Society (Industry Fellowship), Wellcome Trust (205067/Z/16/Z), Alan Turing Institute and the Alzheimer's Drug Discovery Foundation. CBS acknowledges support from the Philip Leverhulme Prize, the Royal Society Wolfson Fellowship, the EPSRC grants EP/S026045/1 and EP/T003553/1, EP/N014588/1, EP/T017961/1, the Wellcome Innovator Award RG98755, the Leverhulme Trust project Unveiling the invisible, the European Union Horizon 2020 research and innovation programme under the Marie Skodowska-Curie grant agreement No. 777826 NoMADS, the Cantab Capital Institute for the Mathematics of Information and the Alan Turing Institute.

Chapter 4

Non-Uniform Diffusion Models

Diffusion models have emerged as one of the most promising frameworks for deep generative modeling. In this work, we explore the potential of non-uniform diffusion models. We show that non-uniform diffusion leads to multi-scale diffusion models which have similar structure to this of multi-scale normalizing flows. We experimentally find that in the same or less training time, the multi-scale diffusion model achieves better FID score than the standard uniform diffusion model. More importantly, it generates samples 4.4 times faster in 128×128 resolution. The speed-up is expected to be higher in higher resolutions where more scales are used. Moreover, we show that non-uniform diffusion leads to a novel estimator for the conditional score function which achieves on par performance with the state-of-the-art conditional denoising estimator. Our theoretical and experimental findings are accompanied by an open source library MSDiff which can facilitate further research of non-uniform diffusion models.

4.1 Introduction

The goal of generative modelling is to learn a probability distribution from a finite set of samples. This classical problem in statistics has been studied for many decades, but until recently efficient learning of high-dimensional distributions remained impossible in practice. For images, the strong inductive biases of convolutional neural networks have recently enabled the modelling of such distributions, giving rise to the field of deep generative modelling.

Deep generative modelling became one of the central areas of deep learning with many successful applications. In recent years much progress has been made in unconditional and conditional image generation. The most prominent approaches are auto-regressive

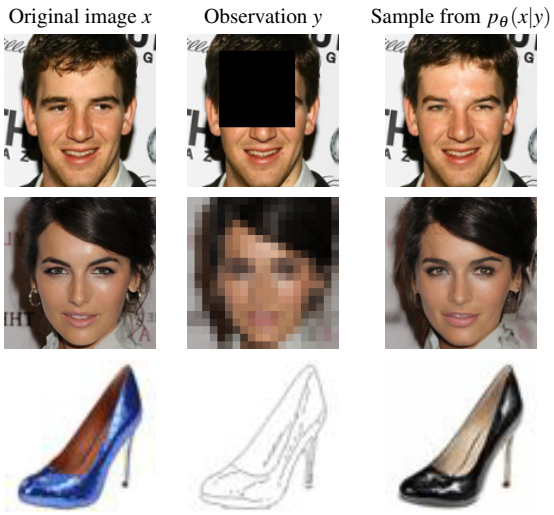


Fig. 4.1 Results from our conditional multi-speed diffusive estimator.

models [18], variational auto-encoders (VAEs) [103], normalizing flows [147] and generative adversarial networks (GANs) [62].

Despite their success, each of the above methods suffers from important limitations. Auto-regressive models allow for likelihood estimation and high-fidelity image generation, but suffer from poor time complexity in high resolutions. VAEs and normalizing flows are less computationally expensive and allow for likelihood estimation, but tend to produce samples of lower visual quality. Moreover, normalizing flows put restrictions on the possible model architectures (requiring invertibility of the network and a Jacobian log-determinant that is computationally tractable), thus limiting their expressivity. While GANs produce state-of-the art quality samples, they don't allow for likelihood estimation and are notoriously hard to train due to training instabilities and mode collapse.

Recently, score-based [84] and diffusion-based [177] generative models have been revived and improved in [182] and [77]. The connection between the two frameworks in discrete-time formulation has been discovered in [206]. Recently in [186], both frameworks have been unified into a single continuous-time approach based on stochastic differential equations [186] and are called score-based diffusion models. These approaches have recently received a lot of attention, achieving state-of-the-art performance in likelihood estimation [186] and unconditional image generation [42], surpassing even the celebrated success of GANs.

In addition to achieving state-of-the art performance in both image generation and likelihood estimation, score-based diffusion models don't suffer from training instabilities or mode collapse [42, 186]. However, although their time complexity in high resolutions is better than that of auto-regressive models [42], it is still notably worse to that of GANs, normalizing

flows and VAEs. Despite the recent efforts to close the sampling time gap between diffusion models and the faster frameworks, diffusion models still require significantly more time to achieve equal performance.

In this work, we explore non-uniform diffusion models. In non-uniform diffusion models, different parts of the input tensor diffuse with different diffusion speeds or more generally according to different stochastic differential equations. We find that the generalization of the original uniform diffusion framework can lead to multi-scale diffusion models which achieve improved sampling performance at a significantly faster sampling speed.

Moreover, we find that non-uniform diffusion can be used for conditional generation, because it leads to a novel estimator of the conditional score. We conduct a review and classification of existing approaches and perform a systematic comparison to find the best way of estimating the conditional score. We provide a proof of validity for the *conditional denoising estimator* (which has been used in [167, 196] without justification), and we thereby provide a firm theoretical foundation for using it in future research.

The contributions of this paper are as follows:

1. We introduce a principled objective for training non-uniform diffusion models.

2. We show that non-uniform diffusion leads to the multi-scale diffusion models which are more efficient than uniform diffusion models. In less training time, the multi-scale models reach improved FID scores with significantly faster sampling speed. The speed up factor is expected to increase as we increase the number of scales.

3. We show that non-uniform diffusion leads to *conditional multi-speed diffusive estimator* (CMDE), a novel estimator of conditional score, which unifies previous methods of conditional score estimation.

4. We provide a proof of consistency for the *conditional denoising estimator* - one of the most successful approaches to estimating the conditional score.

5. We review and empirically compare score-based diffusion approaches to modelling conditional distributions of image data. The models are evaluated on the tasks of super-resolution, inpainting and edge to image translation.

6. We provide an open-source library MSDiff, to facilitate further research on conditional and non-uniform diffusion models.

4.2 Notation

In this work we will use the following notation:

- **Functions of time**

$$f_t := f(t)$$

- **Indexing vectors**
  Let $v = (v_1, ..., v_n) \in \mathbb{R}^n$ and let $1 \leq i < j < n$. Then:

$$v[:j] := (v_1, v_2, ..., v_j) \in \mathbb{R}^j,$$

  cf. Section 4.3.4.

- **Probability distributions**
  We denote the probability distribution of a random variable solely via the name of its density's argument, e.g.

$$p(x_t) := p_{X_t}(x_t),$$

  where x_t is a realisation of the random variable X_t.

- **Iterated Expectations**

$$\begin{aligned}&\mathbb{E}_{\substack{z_1 \sim p(z_1) \\ \vdots \\ z_n \sim p(z_n)}}[f(z_1, \cdots, z_n)] \\ :=&\mathbb{E}_{z_1 \sim p(z_1)} \cdots \mathbb{E}_{z_n \sim p(z_n)}[f(z_1, \cdots, z_n)]\end{aligned}$$

4.3 Methods

In the following, we will provide details about the framework and estimators discussed in this paper.

4.3.1 Background: Score matching through Stochastic Differential Equations

Score-Based Diffusion

In a recent work [186] score-based [84, 182] and diffusion-based [177, 77] generative models have been unified into a single continuous-time score-based framework where the diffusion is driven by a stochastic differential equation. This framework relies on Anderson's Theorem [2], which states that under certain Lipschitz conditions on $f : \mathbb{R}^{n_x} \times \mathbb{R} \to \mathbb{R}^{n_x}$ and $G : \mathbb{R}^{n_x} \times \mathbb{R} \to \mathbb{R}^{n_x} \times \mathbb{R}^{n_x}$ and an integrability condition on the target distribution $p(\mathbf{x}_0)$ a forward diffusion process governed by the following SDE:

$$d\mathbf{x}_t = f(\mathbf{x}_t, t)\, dt + G(\mathbf{x}_t, t)\, d\mathbf{w}_t \tag{4.1}$$

has a reverse diffusion process governed by the following SDE:

$$d\mathbf{x}_t = \left[f(\mathbf{x}_t, t) - G(\mathbf{x}_t, t) G(\mathbf{x}_t, t)^T \nabla_{\mathbf{x}_t} \ln p_{\mathbf{X}_t}(\mathbf{x}_t) \right] dt + G(\mathbf{x}_t, t)\, d\bar{\mathbf{w}}_t \tag{4.2}$$

where $\bar{\mathbf{w}}_t$ is a standard Wiener process in reverse time.

The forward diffusion process transforms the *target distribution* $p(\mathbf{x}_0)$ to a *diffused distribution* $p(\mathbf{x}_T)$ after diffusion time T. By appropriately selecting the drift and the diffusion coefficients of the forward SDE, we can make sure that after sufficiently long time T, the diffused distribution $p(\mathbf{x}_T)$ approximates a simple distribution, such as $\mathcal{N}(\mathbf{0}, \mathbf{I})$. We refer to this simple distribution as the *prior distribution*, denoted by π. The reverse diffusion process transforms the diffused distribution $p(\mathbf{x}_T)$ to the data distribution $p(\mathbf{x}_0)$ and the prior distribution π to a distribution p^{SDE}. p^{SDE} is close to $p(\mathbf{x}_0)$ if the diffused distribution $p(\mathbf{x}_T)$ is close to the prior distribution π. We get samples from p^{SDE} by sampling from π and simulating the reverse sde from time T to time 0.

To get samples by simulating the reverse SDE, we need access to the time-dependent score function $\nabla_{\mathbf{x}_t} \ln p(\mathbf{x}_t)$ for all $\mathbf{x}_t$ and t. In practice, we approximate the time-dependent score function with a neural network $s_\theta(\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}_t} \ln p(\mathbf{x}_t)$ and simulate the reverse SDE in equation 4.3 to map the prior distribution π to p_θ^{SDE}.

$$d\mathbf{x}_t = \left[f(\mathbf{x}_t, t) - G(\mathbf{x}_t, t) G(\mathbf{x}_t, t)^T s_\theta(\mathbf{x}_t, t) \right] dt + G(\mathbf{x}_t, t)\, d\bar{\mathbf{w}}_t \tag{4.3}$$

If the prior distribution is close to the diffused distribution and the approximated score function is close to the ground truth score function, the modeled distribution p_θ^{SDE} is provably

close to the target distribution $p(\mathbf{x}_0)$. This statement is formalised in the language of distributional distances in the next subsection.

Uniform Diffusion Models

Previous works [77, 182, 42] used the same forward SDE for the diffusion of all the pixels. For this reason, we classify them as uniform diffusion models. In uniform diffusion models, the sde in equation 4.4 describes the forward diffusion for all pixels in an image:

$$dx_t = f(x_t,t)dt + g(t)d\bar{w}_t, \tag{4.4}$$

We used unbold notation for the random variables to show that this equation describes diffusion in one dimension. For uniform diffusion models, the neural network $s_\theta(\mathbf{x}_t,t)$ can be trained to approximate the score function $\nabla_{\mathbf{x}_t}\ln p(\mathbf{x}_t)$ by minimizing the weighted score matching objective

$$\mathscr{L}_{SM}(\theta,\lambda(\cdot)) := \frac{1}{2}\mathbb{E}_{\substack{t\sim U(0,T)\\ \mathbf{x}_t\sim p(\mathbf{x}_t)}}[\lambda(t)\,\|\nabla_{\mathbf{x}_t}\ln p(\mathbf{x}_t) - s_\theta(\mathbf{x}_t,t)\|_2^2] \tag{4.5}$$

where $\lambda : [0,T] \to \mathbb{R}_+$ is a positive weighting function.

However, the above quantity cannot be optimized directly since we don't have access to the ground truth score $\nabla_{\mathbf{x}_t}\ln p(\mathbf{x}_t)$. Therefore in practice, a different objective has to be used [84, 182, 186]. In [186], the weighted denoising score-matching objective is used, which is defined as

$$\mathscr{L}_{DSM}(\theta,\lambda(\cdot)) := \frac{1}{2}\mathbb{E}_{\substack{t\sim U(0,T)\\ \mathbf{x}_0\sim p(\mathbf{x}_0)\\ \mathbf{x}_t\sim p(\mathbf{x}_t|\mathbf{x}_0)}}\left[\lambda(t)\,\|\nabla_{\mathbf{x}_t}\ln p(\mathbf{x}_t|\mathbf{x}_0) - s_\theta(\mathbf{x}_t,t)\|_2^2\right] \tag{4.6}$$

The difference between DSM and SM is the replacement of the ground truth score which we do not know by the score of the perturbation kernel which we know analytically for many choices of forward SDEs. The choice of the weighted DSM objective is justified because the weighted DSM objective is equal to the SM objective up to a constant that does not depend on the parameters of the model θ. The reader can refer to [206] for the proof.

The choice of the weighting function is also important, because it determines the quality of score-matching in different diffusion scales. A principled choice for the weighting function is $\lambda(t) = g(t)^2$, where $g(\cdot)$ is the diffusion coefficient of the forward SDE. This weighting function is called the likelihood weighting function [181], because it ensures that we minimize an upper bound on the Kullback–Leibler divergence from the target distribution to the model distribution by minimizing the weighted DSM objective with this weighting.

The previous statement is implied by the combination of inequality 4.7 which is proven in [181] and the relationship between the DSM and SM objectives.

$$D_{KL}(p(\mathbf{x}_0) \parallel p_\theta^{SDE}) \leq L_{SM}(\theta, g(\cdot)^2) + D_{KL}(p(\mathbf{x}_T) \parallel \pi) \tag{4.7}$$

Other weighting functions have also yielded very good results with particular choices of forward sdes. However, we do not have theoretical guarantees that alternative weightings would yield good results with arbitrary choices of forward sdes.

4.3.2 Non-Uniform Diffusion Models

In this section, we describe non-uniform diffusion models. We call them non-uniform to indicate that the forward diffusion of each pixel is potentially governed by a different SDE. Considering a vectorised form $x = \text{vec}(X) = [x^1, x^2, ..., x^{mnc}]$ of an image $X \in [0,1]^m \times [0,1]^n \times [0,1]^c$, we assume that the diffusion of the i^{th} pixel is governed by the following SDE:

$$dx_t^i = f_i(x_t^i, t)dt + g_i(t)dw_t^i \tag{4.8}$$

Equation 4.8 is a special case of the general Itô SDE described in equation 4.1, but provides more flexibility compared to uniform diffusion where all pixels diffuse independently according to the same SDE. The diffusion of the entire image vector is summarised by the following SDE:

$$d\mathbf{x}_t = f(\mathbf{x}_t, t)dt + G(t)d\mathbf{w}_t, \tag{4.9}$$

where $f(\mathbf{x}_t) = [f_1(x^1, t), ..., f_{mnc}(x_t^{mnc}, t)]$ and $G(t) = \text{diag}([g_1(t), ..., g_{mnc}(t)])$.

In this more general setup, the DSM objective as described in equation 4.6 must also be generalised. The positive weighting function $\lambda(\cdot)$ is replaced by a positive definite matrix $\Lambda(\cdot)$ which gives the form of the DSM objective for non-uniform diffusion models:

$$\mathscr{L}_{DSM}(\theta, \Lambda(\cdot)) := \frac{1}{2}\mathbb{E}_{\substack{t\sim U(0,T) \\ \mathbf{x}_0 \sim p(\mathbf{x}_0) \\ \mathbf{x}_t \sim p(\mathbf{x}_t|\mathbf{x}_0)}} \left[\mathbf{v}_\theta(\mathbf{x}_0, \mathbf{x}_t, t)^T \Lambda(t) \mathbf{v}_\theta(\mathbf{x}_0, \mathbf{x}_t, t)\right] \tag{4.10}$$

where $\mathbf{v}_\theta(\mathbf{x}_0, \mathbf{x}_t, t) = \nabla_{\mathbf{x}_t} \ln p(\mathbf{x}_t|\mathbf{x}_0) - s_\theta(\mathbf{x}_t, t)$

We prove that a principled choice for the positive weighting matrix is $\Lambda_{MLE}(t) = G(t)G(t)^T$. We call it the likelihood weighting matrix for non-uniform diffusion because it ensures minimization of an upper bound to the KL divergence from the target distribution

to the model distribution. The previous statement is summarised in Theorem 4.3.1 which is proved in section B.1.3 of the Appendix.

Theorem 4.3.1. *Let $p(\mathbf{x}_t)$ denote the distribution implied by the forward SDE at time t and $p_\theta^{SDE}(\mathbf{x}_t)$ denote the distribution implied by the parametrized reverse SDE at time t. Then under regularity assumptions of [181, Theorem 1], we have that*

$$KL(p(\mathbf{x}_0)\|p_\theta^{SDE}(\mathbf{x}_0)) \leq KL(p(\mathbf{x}_T)\|\pi(\mathbf{x}_T)) + \frac{1}{2}\mathbb{E}_{\substack{t\sim U(0,T)\\ \mathbf{x}_t\sim p(\mathbf{x}_t)}}\left[\mathbf{v}^T G(t)G(t)^T\mathbf{v}\right],$$

where $\mathbf{v} = \nabla_{\mathbf{x}_t}\ln p(\mathbf{x}_t) - s_\theta(\mathbf{x}_t,t)$.

4.3.3 Application of Non-Uniform Diffusion in multi-scale diffusion

We design the forward process so that different groups of pixels diffuse with different speeds which creates a multi-scale diffusion structure. The intuition stems from multi-scale normalising flows. Multi-scale normalising flows invertibly transform the input tensor to latent encodings of different scales by splitting the input tensor into two parts after transformation in each scale. The multi-scale structure in normalising flows is shown to lead to faster training and sampling without compromise in generated image quality.

We intend to transfer this idea to score-based modeling by diffusing some parts of the tensor faster. There are many ways to split the image into different parts which diffuse faster. It has been experimentally discovered that cascaded diffusion models [167] yield improved results compared to standard diffusion models. This gave us the intuition to use a multi-level haar transform to transform every image to n high frequency scales $d_1,...,d_n$ (detail coefficients) and one low frequency scale a_n (approximation coefficient). The natural generation order of the haar coefficients (in line with cascaded diffusion) is a_n, d_n, d_{n-1}, ..., d_1. For this reason, we choose to diffuse lower frequency coefficients slower than high frequency coefficients. More specifically, we design the forward process so that all coefficients reach the same signal-to-noise ratio at the end of their diffusion time. We set the diffusion time for a_n to $T_{a_n} = 1$ and for d_i to $T_{d_i} = \frac{i}{n+1}$ for each $i \in [1,...,n]$. We illustrate the multiscale model structure graphically in Figure 4.2.

Training

We approximate the score of the distribution of $c_i(t) = [a_n(t), d_n(t), ..., d_i(t)]$ in the time range $[(i-1)/(n+1), i/(n+1)]$ with a separate neural network $s_i(c_i(t),t)$. We also use a separate network $s_{n+1}(c_{n+1}(t),t)$ to approximate the score of the distribution of $c_{n+1}(t) = a_n(t)$ in the diffusion time range $[n/(n+1), 1]$. We use different networks in each scale to leverage

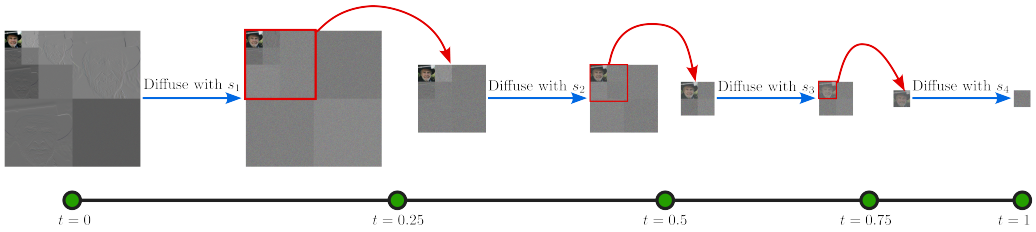


Fig. 4.2 Model with three scales. d_1 reaches the target SNR at $t = 0.25$ and does not diffuse further. The remaining part of tensor continues the diffusion. The diffusion procedure is continued as implied until a_3 reaches the target SNR at time $t = 1$. We use four neural networks S_1, S_2, S_3, S_4 to approximate the score function in the intermediate diffusion intervals, because the dimensionality of the diffusing tensor decreases every time some part of the tensor reaches the target SNR.

the fact that we approximate the score of lower dimensional distributions. This enables faster score function evaluation and, therefore, faster training and sampling. We train each network separately using the likelihood weighting matrix for non-uniform diffusion (see section 4.3.2).

Sampling

The sampling process is summarised in the following steps:

1. Sample $a_n(1)$ from the stationary distribution (e.g. standard normal distribution) and integrate the reverse sde for a_n from time $t = 1$ to time $t = n/(n+1)$.
2. Sample $d_n(n/(n+1))$ from the stationary distribution and solve the reverse sde for $[a_n, d_n]$ from time $t = n/(n+1)$ to time $t = (n-1)/(n+1)$.
3. The process is continued as implied until we reach the final generation level, where we sample $d_1(1/(n+1))$ from the stationary distribution and solve the reverse sde for $[a_n, d_n, ..., d_1]$ from time $t = 1/(n+1)$ to time $t = \varepsilon$ (e.g., $\varepsilon = 10^{-5}$).
4. We convert the generated haar coefficients $[a_n(\varepsilon), d_n(\varepsilon), ..., d_1(\varepsilon)]$ to the generated image using the multi-level inverse haar transform.

Our experimental results presented in section 4.4.1 show that multiscale diffusion is more efficient and effective than uniform diffusion.

4.3.4 Application of Non-Uniform Diffusion in Conditional generation

The continuous score-matching framework can be extended to conditional generation, as shown in [186]. Suppose we are interested in $p(x|y)$, where x is a *target image* and y is a *condition image*. Again, we use the forward diffusion process (Equation 4.1) to obtain a family of diffused distributions $p(x_t|y)$ and apply Anderson's Theorem to derive the *conditional reverse-time SDE*

$$dx = [\mu(x,t) - \sigma(t)^2 \nabla_x \ln p_{X_t}(x|y)]dt + \sigma(t)d\tilde{w}. \tag{4.11}$$

Now we need to learn the score $\nabla_{x_t} \ln p(x_t|y)$ in order to be able to sample from $p(x|y)$ using reverse-time diffusion.

In this work, we discuss the following approaches to estimating the conditional score $\nabla_{x_t} \ln p(x_t|y)$:

1. Conditional denoising estimators
2. Conditional diffusive estimators
3. Multi-speed conditional diffusive estimators (our method)

We discuss each of them in a separate section.

In [186] an additional approach to conditional score estimation was suggested: This method proposes learning $\nabla_{x_t} \ln p(x_t)$ with an unconditional score model, and learning $p(y|x_t)$ with an auxiliary model. Then, one can use

$$\nabla_{x_t} \ln p(x_t|y) = \nabla_{x_t} \ln p(x_t) + \nabla_{x_t} \ln p(y|x_t)$$

to obtain $\nabla_{x_t} \ln p(x_t|y)$. Unlike other approaches, this requires training a separate model for $p(y|x_t)$. Appropriate choices of such models for tasks discussed in this paper have not been explored yet. Therefore we exclude this approach from our study.

Conditional denoising estimator (CDE)

The conditional denoising estimator (CDE) is a way of estimating $p(x_t|y)$ using the denoising score matching approach [206, 182]. In order to approximate $p(x_t|y)$, the conditional

denoising estimator minimizes

$$\frac{1}{2}\mathbb{E}_{\substack{t\sim U(0,T)\\ x_0,y\sim p(x_0,y)\\ x_t\sim p(x_t|x_0)}}[\lambda(t)\,\|\nabla_{x_t}\ln p(x_t|x_0)-s_\theta(x_t,y,t)\|_2^2] \tag{4.12}$$

This estimator has been shown to be successful in previous works [167, 196], also confirmed in our experimental findings (cf. Section 4.4).

Despite the practical success, this estimator has previously been used without a theoretical justification of why training the above objective yields the desired conditional distribution. Since $p(x_t|y)$ does not appear in the training objective, it is not obvious that the minimizer approximates the correct quantity.

By extending the arguments of [206], we provide a formal proof that the minimizer of the above loss does indeed approximate the correct conditional score $p(x_t|y)$. This is expressed in the following theorem.

Theorem 4.3.2. *The minimizer (in θ) of*

$$\frac{1}{2}\mathbb{E}_{\substack{t\sim U(0,T)\\ x_0,y\sim p(x_0,y)\\ x_t\sim p(x_t|x_0)}}[\lambda(t)\,\|\nabla_{x_t}\ln p(x_t|x_0)-s_\theta(x_t,y,t)\|_2^2]$$

is the same as the minimizer of

$$\frac{1}{2}\mathbb{E}_{\substack{t\sim U(0,T)\\ x_t,y\sim p(x_t,y)}}[\lambda(t)\,\|\nabla_{x_t}\ln p(x_t|y)-s_\theta(x_t,y,t)\|_2^2]$$

The proof for this statement can be found in Appendix B.1.1. Using the above theorem, the consistency of the estimator can be established.

Corollary 4.3.3. *Let θ^* be a minimizer of a Monte Carlo approximation of (4.12), then (under technical assumptions, cf. Appendix B.1.2) the conditional denoising estimator $s_{\theta^*}(x,y,t)$ is a consistent estimator of the conditional score $\nabla_{x_t}\ln p(x_t|y)$, i.e.*

$$s_{\theta^*}(x,y,t)\xrightarrow{P}\nabla_{x_t}\ln p(x_t|y)$$

as the number of Monte Carlo samples approaches infinity.

This follows from the previous theorem and the uniform law of large numbers. Proof in the Appendix B.1.2.

Conditional diffusive estimator (CDiffE)

Conditional diffusive estimators (CDiffE) have first been suggested in [186]. The core idea is that instead of learning $p(x_t|y)$ directly, we diffuse both x and y and approximate $p(x_t|y_t)$, using the denoising score matching. Just like learning diffused distribution $\nabla_{x_t} \ln p(x_t)$ improves upon direct estimation of $\nabla_x \ln p(x)$ [182, 186], diffusing both the input x and condition y, and then learning $\nabla_{x_t} \ln p(x_t|y_t)$ could make optimization easier and give better results than learning $\nabla_{x_t} \ln p(x_t|y)$ directly.

In order to learn $p(x_t|y_t)$, observe that

$$\nabla_{x_t} \ln p(x_t|y_t) = \nabla_{x_t} \ln p(x_t, y_t) = \nabla_{z_t} \ln p(z_t)[:n_x],$$

where $z_t := (x_t, y_t)$ and n_x is the dimensionality of x. Therefore we can learn the (unconditional) score of the joint distribution $p(x_t, y_t)$ using the denoising score matching objective just like as in the unconditional case, i.e

$$\frac{1}{2}\mathbb{E}_{\substack{t\sim U(0,T) \\ z_0\sim p_0(z_0) \\ z_t\sim p(z_t|z_0)}} [\lambda(t) \|\nabla_{z_t} \ln p(z_t|z_0) - s_\theta(z_t, t)\|_2^2]. \tag{4.13}$$

We can then extract our approximation for the conditional score $\nabla_{x_t} \ln p(x_t|y_t)$ by simply taking the first n_x components of $s_\theta(x_t, y_t, t)$.

The aim is to approximate $\nabla_{x_t} \ln p(x_t|y)$ with $\nabla_{x_t} \ln p(x_t|\hat{y}_t)$, where $\hat{y}_t$ is a sample from $p(y_t|y)$. Of course this approximation is imperfect and introduces an error, which we call the *approximation error*. CDiffE aims to achieve smaller optimization error by diffusing the condition y and making the optimization landscape easier, at a cost of making this approximation error.

Now in order to obtain samples from the conditional distribution, we sample a point $x_T \sim \pi$ and integrate

$$dx = [\mu(x,t) - \sigma(t)^2 \nabla_x \ln p_{X_t|Y_t}(x|\hat{y}_t)]dt + \sigma(t)d\tilde{w}$$

from T to 0, sampling $\hat{y}_t \sim p(y_t|y)$ at each time step.

Conditional multi-speed diffusive estimator (CMDE)

In this section we present a novel estimator for the conditional score $\nabla_{x_t} \ln p(x_t|y)$ which we call the *conditional multi-speed diffusive estimator* (CMDE).

Sources of error for different estimators

CDE
Optimization error:

$$s_\theta(x,y,t) \approx \nabla_{x_t} \ln p(x_t|y)$$

CDiffE and CMDE
Optimization error:

$$s_\theta(x,y,t) \approx \nabla_{x_t} \ln p(x_t|y_t)$$

Approximation error:

$$\nabla_{x_t} \ln p(x_t|\hat{y}_t) \approx \nabla_{x_t} \ln p(x_t|y)$$

CDiffE aims to achieve smaller optimization error at a cost of higher approximation error. By controlling the diffusion speed of y, CMDE tries to find an optimal balance between optimization error and approximation error.

Fig. 4.3 Sources of error for different estimators

Our approach is based on two insights. Firstly, there is no reason why x_t and y_t in conditional diffusive estimation need to diffuse at the same rate. Secondly, by decreasing the diffusion rate of y_t while keeping the diffusion speed of x_t the same, we can bring $p(x_t|y_t)$ closer to $p(x_t|y)$, at the possible cost of making the optimization more difficult. This way we can *interpolate* between the conditional denoising estimator and the conditional diffusive estimator and find an optimal balance between optimization error and approximation error (cf. Figure 4.3). This can lead to a better performance, as indicated by our experimental findings (cf. Section 4.4).

In our conditional multi-speed diffusive estimator, x_t and y_t diffuse according to SDEs with the same drift but different diffusion rates,

$$dx = \mu(x,t)dt + \sigma^x(t)dw$$
$$dy = \mu(y,t)dt + \sigma^y(t)dw.$$

Then, just like in the case of conditional diffusive estimator, we try to approximate the joint score $\nabla_{x_t,y_t} \ln p(x_t,y_t)$ with a neural network. Since x_t and y_t now diffuse according to different SDEs, we need to take this into account and replace the weighting function $\lambda(t) : \mathbb{R} \to \mathbb{R}_+$ with a positive definite weighting matrix $\Lambda(t) : \mathbb{R} \to \mathbb{R}^{(n_x+n_y)\times(n_x+n_y)}$. Hence,

the new training objective becomes

$$\frac{1}{2}\mathbb{E}_{\substack{t\sim U(0,T)\\ z_0\sim p_0(z_0)\\ z_t\sim p(z_t|z_0)}}[v^T\Lambda(t)v], \tag{4.14}$$

where $v = \nabla_{z_t}\ln p(z_t|z_0) - s_\theta(z_t,t)$, $z_t = (x_t, y_t)$.

In [181] authors derive a likelihood weighting function $\lambda^{\text{MLE}}(t)$, which ensures that the objective of the score-based model upper-bounds the negative log-likelihood of the data, thus enabling approximate maximum likelihood training of score-based diffusion models. We generalize this result to the multi-speed diffusion case by providing a likelihood weighting matrix $\Lambda^{\text{MLE}}(t)$ with the same properties.

Theorem 4.3.4. *Let $\mathscr{L}(\theta)$ be the CMDE training objective (Equation 4.14) with the following weighting:*

$$\Lambda^{MLE}_{i,j}(t) = \begin{cases} \sigma^x(t)^2, \text{ if } i = j,\ i \le n_x \\ \sigma^y(t)^2, \text{ if } i = j,\ n_x < i \le n_y \\ 0, \text{ otherwise} \end{cases}$$

Then the joint negative log-likelihood is upper bounded (up to a constant in θ) by the training objective of CMDE

$$-\mathbb{E}_{(x,y)\sim p(x,y)}[\ln p_\theta(x,y)] \le \mathscr{L}(\theta) + C.$$

The proof can be found in Appendix B.1.3.

Moreover we show that the mean squared approximation error of a multi-speed diffusion model is upper bounded and the upper bound goes to zero as the diffusion speed of the condition $\sigma^y(t)$ approaches zero.

Theorem 4.3.5. *Fix t, x_t and y. Under mild technical assumptions (cf. Appendix B.1.4) there exists a function $E : \mathbb{R} \to \mathbb{R}$ monotonically decreasing to 0, such that*

$$\begin{aligned}\mathbb{E}_{y_t\sim p(y_t|y)}[\|\nabla_{x_t}\ln p(x_t|y_t) - \nabla_{x_t}\ln p(x_t|y)\|_2^2] \\ \le E(1/\sigma^y(t)).\end{aligned}$$

The proof can be found in Appendix B.1.4.

Thus we see that the objective of CMDE approaches that of CDE as $\sigma^y(t) \to 0$, and CMDE coincides with CDiffE when $\sigma^y(t) = \sigma^x(t)$ (cf. Figure 4.3).

We experimented with different configurations of $\sigma^x(t)$ and $\sigma^y(t)$ and found configurations that lead to improvements upon CDiffE and CDE in certain tasks. The experimental results are discussed in detail in Section 4.4.2.

4.4 Experiments

4.4.1 Multiscale diffusion

In this part of the experimental section, we compare the performance of the multiscale model that depends on non-uniform pixel diffusion to the performance of the standard model that depends on uniform diffusion. We train and evaluate both models on CelebA-HQ 128×128.

For the standard diffusion model, we use the beta-linear VP SDE [77] and train the score model using the simple objective [42] because it is experimentally shown to favor generation quality. The architecture of the score model follows the architecture of [42].

For the multiscale model, we use 3-level haar transform to transform the original images, which means that we create a multiscale model with four scales. For this reason, we use four score models $s_{\theta_1}, s_{\theta_2}, s_{\theta_3}, s_{\theta_4}$ which approximate the score function in following diffusion ranges respectively $[\varepsilon, 0.25], [0.25, 0.50], [0.50, 0.75], [0.75, 1]$. The reason we do not use s_{θ_1} to approximate the score function for the entire diffusion is that we stop the diffusion of the highest frequency detail coefficients d_1 at time 0.25, as they reach the target minimum SNR (by design of the forward SDE). The remaining diffusing tensor has a quarter of the dimensionality of the original tensor. Therefore, we need a less expressive neural network to approximate the score function in the next diffusion time range. The architecture of all models follows the architecture of [42]. We choose the number of base channels and the depth of the multiscale score models so that the total number of parameters of the multiscale model is approximately equal to the number of parameters of the standard diffusion model to ensure fair comparison. For the diffusion of each haar coefficient, we use a variance preserving process with log-linear SNR profile as in [101]. We choose the maximum SNR (at $t = \varepsilon$) and the minimum SNR (achieved at the terminal diffusion time for each coefficient) to match the maximum SNR and minimum SNR of the standard model respectively.

We evaluate both models using the FID score on 50K samples. We generate each sample by numerically integrating the reverse SDE with 256 total euler-maruyama steps and provide qualitative results at the Appendix B.3. Our results (see Tables 4.1) show that for the same training time, the multiscale model achieves better FID score with significantly faster sampling speed (4.4 times faster). In fact, our results show that the multiscale model achieves improved FID score with faster sampling speed and less training time. We verified that by

integrating the corresponding probability flow ODEs using the euler method. In that case, we got lower FID scores for both methods but the relative performance remained the same. Moreover, we used lighter neural networks than prior works to approximate the score function which led to generally worse performance. We opted for lighter models in this study because we wanted to conduct a fair comparison of the multiscale diffusion model and the standard uniform diffusion model. Improved techniques that led to state-of-the-art performance of the uniform diffusion model such as class conditioning and learning of the variance schedule [42] can also be readily employed in the multiscale model. Given the superiority of the multiscale model, we expect the employment of improved techniques to further improve the performance of the multiscale model and potentially redefine the state-of-the-art. We intend to explore this direction in the future.

The training and sampling speed-up is attributed to the fact that we approximate the score of lower dimensional distributions for the majority of the diffusion. Therefore, we expect higher relative speed-ups in higher resolutions. We believe that the effectiveness of the multiscale model is attributed to the effectiveness of cascaded diffusion observed in previous works [167, 42]. The difference between our multiscale model and the previous works is that it does not suffer from the effect of the compounding error. Ho et al. [167] improve the performance of cascading models by using an expensive post-training tuning step which they call conditioning augmentation. Our multiscale model essentially employs a cascading modeling structure that does not require any post-training tuning for improved sample generation.

Table 4.1 Multiscale and Vanilla model comparison on CelebA-HQ 128x128

	Iterations	Parameters	Training (hours) ↓	Sampling (secs) ↓	FID ↓
Vanilla	0.67M	100M	128	53.5	54.3
Multiscale	2.54M	100M	128	12.1	31.8
Multiscale +	1.76M	200M	128	18.7	33.5

4.4.2 Conditional Generation

In this section we conduct a systematic comparison of different score-based diffusion approaches to modelling conditional distributions of image data. We evaluate these approaches on the tasks of super-resolution, inpainting and edge to image translation.

Original image x Observation y Reconstruction $\hat{x}_1$ Reconstruction $\hat{x}_2$ Reconstruction $\hat{x}_3$ Reconstruction $\hat{x}_4$ Reconstruction $\hat{x}_5$

Fig. 4.4 Diversity of five different CMDE reconstructions for a given masked image.

Datasets In our experiments, we use the CelebA [123] and Edges2shoes [217, 87] datasets. We pre-processed the CelebA dataset as in [119].

Models and hyperparameters In order to ensure the fair comparison, we separate the evaluation of a particular estimator of conditional score from the evaluation of a particular neural network model. To this end, we train the same neural network architecture for all estimators. The architecture is based on the DDPM model used in [77, 186]. We used the variance-exploding SDE [186] given by:

$$dx = \sqrt{\frac{d}{dt}\sigma^2(t)}dw, \qquad \sigma(t) = \sigma_{min}\left(\frac{\sigma_{max}}{\sigma_{min}}\right)^t$$

Likelihood weighting was employed for all experiments. For CMDE, the diffusion speed of y was controlled by picking an appropriate $\sigma^y_{\max}$, which we found by trial-and-error. The performance of CMDE could be potentially improved by performing a systematic hyperparameter search for optimal $\sigma^y_{\max}$. Details on hyperparameters and architectures used in our experiments can be found in Appendix B.2.

Inverse problems The tasks of inpainting, super-resolution and edge to image translation are special cases of inverse problems [6, 133]. In each case, we are given a (possibly random) forward operator A which maps our data x (full image) to an observation y (masked image, compressed image, sketch). The task is to come up with a high-quality reconstruction $\hat{x}$ of the image x based on an observation y. The problem of reconstructing x from y is typically ill-posed, since y does not contain all information about x. Therefore, an ideal algorithm would produce a reconstruction $\hat{x}$, which looks like a realistic image (i.e. is a likely sample from $p(x)$) and is consistent with the observation y (i.e. $A\hat{x} \approx y$). Notice that if a conditional score model learns the conditional distribution correctly, then our reconstruction $\hat{x}$ is a sample from the posterior distribution $p(x|y)$, which satisfies bespoke requirements. This strategy for solving inverse problems is generally referred to as *posterior sampling*.

Evaluation: Reconstruction quality Ill-posedness often means that we should not strive to reconstruct x perfectly. Nonetheless reconstruction error does correlate with the performance of the algorithm and has been one of the most widely-used metrics in the community. To

evaluate the reconstruction quality for each task, we measure the Peak signal-to-noise ratio (PSNR) [210], Structural similarity index measure (SSIM) [210] and Learned Perceptual Image Patch Similarity (LPIPS) [221] between the original image x and the reconstruction $\hat{x}$.

Evaluation: Consistency In order to evaluate the consistency of the reconstruction, for each task we calculate the PSNR between $y := Ax$ and $\hat{y} := A\hat{x}$.

Evaluation: Diversity We evaluate diversity of each approach by generating five reconstructions $(\hat{x})_{i=1}^5$ for a given observation y. Then for each y we calculate the average standard deviation for each pixel among the reconstructions $(\hat{x})_{i=1}^5$. Finally, we average this quality over 5000 test observations.

Evaluation: Distributional distances If our algorithm generates realistic reconstructions while preserving diversity, then the distribution of reconstructions $p(\hat{x})$ should be similar to the distribution of original images $p(x)$. Therefore, we measure the Fréchet Inception Distance (FID) [73] between unconditional distributions $p(x)$ and $p(\hat{x})$ based on 5000 samples. Moreover, we calculate the FID score between the joint distributions $p(\hat{x}, y)$ and $p(x, y)$, which allows us to simultaneously check the realism of the reconstructions and the consistency with the observation. We use abbreviation UFID to refer to FID between between unconditional distributions and JFID to refer to FID between joints. In our judgement, FID and especially the JFID is the most principled of the used metrics, since it measures how far $p_\theta(x|y)$ is from $p(x|y)$.

Table 4.2 Results of conditional generation tasks.

Task	Estimator	PSNR/SSIM ↑	LPIPS ↓	UFID/JFID ↓	Consistency ↑	Diversity ↑
Inpainting	CDE	**25.12/0.870**	**0.042**	13.07/18.06	**28.54**	4.79
	CDiffE	23.07/0.844	0.057	13.28/19.25	26.61	**6.52**
	CMDE ($\sigma^y_{max} = 1$)	24.92/0.864	0.044	**12.07/17.07**	28.32	4.98
Super-resolution	CDE	23.80/0.650	0.114	10.36/15.77	54.18	**8.51**
	CDiffE	23.83/0.656	0.139	14.29/20.20	51.90	7.41
	CMDE ($\sigma^y_{max} = 0.5$)	23.91/0.654	0.109	**10.28/15.68**	53.03	8.33
	HCFLOW	**24.95/0.702**	**0.107**	14.13/19.55	**55.31**	6.26
Edge to image	CDE	**18.35/0.699**	**0.156**	**11.87/21.31**	**10.45**	14.40
	CDiffE	10.00/0.365	0.350	33.41/55.22	7.78	**43.45**
	CMDE ($\sigma^y_{max} = 1$)	18.16/0.692	0.158	12.62/22.09	10.38	15.20

Inpainting

We perform the inpainting experiment using CelebA dataset. In inpainting, the forward operator A is an application of a given binary mask to an image x. In our case, we made the task more difficult by using randomly placed (square) masks. Then the conditional score model is used to obtain a reconstruction $\hat{x}$ from the masked image y. We select the position

of the mask uniformly at random and cover 25% of the image. The quantitative results are summarised in Table 4.2 and samples are presented in Figure 4.5. We observe that CDE and CMDE significantly outperform CDiffE in all metrics, with CDE having a small advantage over CMDE in terms of reconstruction error and consistency. On the other hand, CMDE achieves the best FID scores.

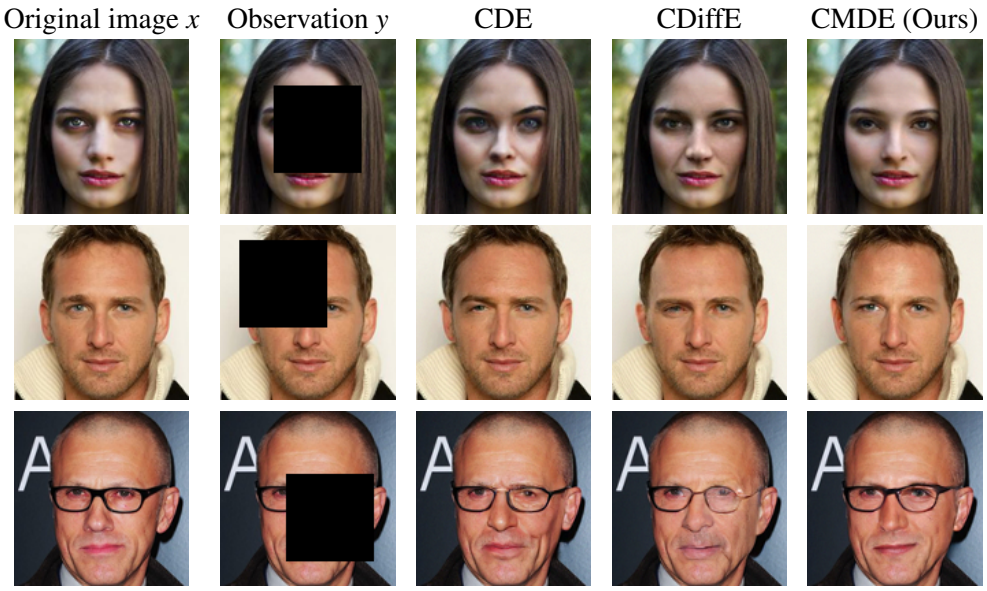


Fig. 4.5 Inpainting results.

Super-resolution

We perform 8x super-resolution using the CelebA dataset. A high resolution 160x160 pixel image x is compressed to a low resolution 20x20 pixels image y. Here we use bicubic downscaling [97] as the forward operator A. Then using a score model we obtain a 160x160 pixel reconstruction image $\hat{x}$. The quantitative results are summarised in Table 4.2 and samples are presented in Figure 4.6. We find that CMDE and CDE perform similarly, while significantly outperforming CDiffE. CMDE achieves the smallest reconstruction error and captures the distribution most accurately according to FID scores.

Edge to image translation

We perform an edge to image translation task on the Edges2shoes dataset. The forward operator A is given by a neural network edge detector [214], which takes an original photo of a shoe x and transforms it into a sketch y. Then a conditional score model is used to create an

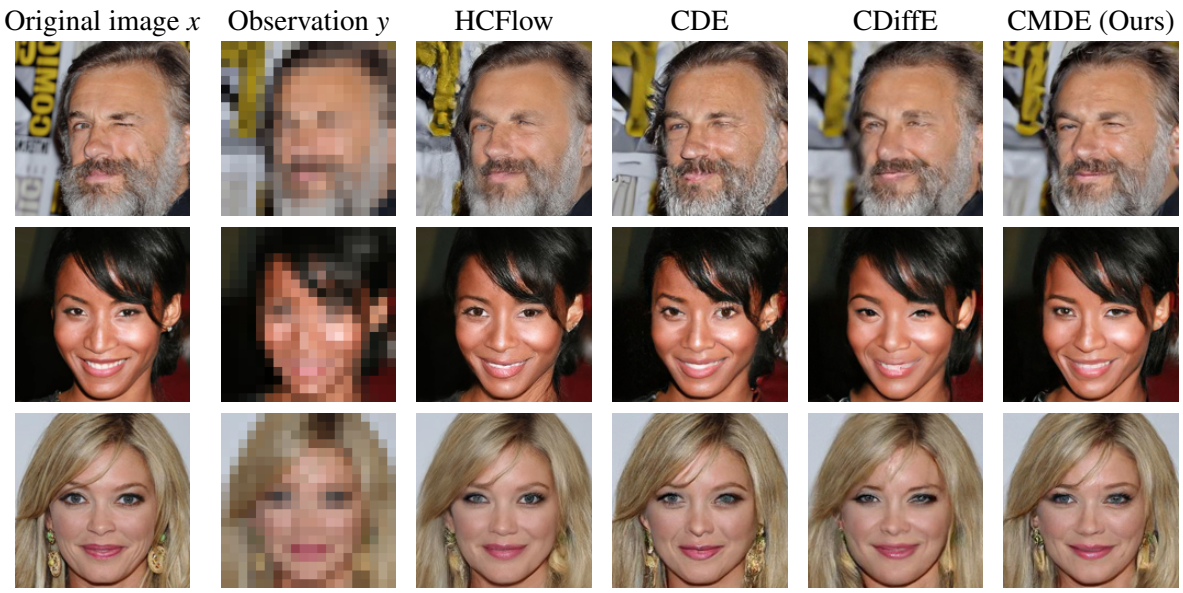


Fig. 4.6 Super-resolution results.

artificial photo of a shoe $\hat{x}$ matching the sketch. The quantitative results are summarised in Table 4.2 and samples are presented in Figure 4.7. Unlike in inpainting and super-resolution where CDiffE achieved reasonable performance, in edge to image translation, it fails to create samples consistent with the condition (which leads to inflated diversity scores). CDE and CMDE are comparable, but CDE performed slightly better across all metrics. However, the performance of CMDE could be potentially improved by tuning the diffusion speed $\sigma^y(t)$.

4.5 Comparison with state-of-the-art

We compare score-based diffusion approaches with HCFlow [119] – a state-of-the-art method in super-resolution. To ensure a fair comparison, we used the data pre-processing and hyperparameters for HCFlow exactly as in the original paper [119]. We find that although HCFlow performs marginally better in terms of the reconstruction error, CDE and CMDE obtain a significantly better FID and diversity scores indicating better distribution coverage. We recall that a perfect reconstruction on a per-image basis is generally not desirable due to ill-posedness of the inverse problem and therefore in our view FID is the most principled of the used metrics. The FID scores suggest that CMDE was the most successful approach to approximating the posterior distribution.

Fig. 4.7 Edge to image translation results.

4.6 Conclusions and future work

In this article, we explored non-uniform diffusion models which rely on the idea of diffusing different parts of the tensor with different speeds or more generally according to different SDEs. We show that non-uniform diffusion leads to multiscale diffusion models which are more efficient and effective than standard uniform diffusion models for unconditional generation. More specifically, multiscale diffusion models achieve improved FID score with significantly faster sampling speed and for less training time.

We further discovered that non-uniform diffusion leads to CMDE, a novel estimator of the conditional score which can interpolate between conditional denoising estimator (CDE) and conditional diffusive estimator (CDiffE). We conducted a systematic comparison of different estimators of the conditional score and concluded that CMDE and CDE perform on par, while significantly outperforming CDiffE. This is particularly apparent in edge to image translation, where CDiffE fails to produce samples consistent with the condition image. Furthermore, CMDE outperformed CDE in terms of FID scores in inpainting and super-resolution tasks, which indicates that diffusing the condition at the appropriate speed can have beneficial effect on the optimization landscape, and yield better approximation of the posterior distribution. Furthermore, we provided theoretical analysis of the estimators of conditional score. More importantly, we proved the consistency of the conditional denoising estimator, thus providing a firm theoretical justification for using it in future research.

4.7 Acknowledgements

GB acknowledges the support from GSK and the Cantab Capital Institute for the Mathematics of Information. JS acknowledges the support from Aviva and the Cantab Capital Institute for the Mathematics of Information. CBS acknowledges support from the Philip Leverhulme Prize, the Royal Society Wolfson Fellowship, the EPSRC advanced career fellowship EP/V029428/1, EPSRC grants EP/S026045/1 and EP/T003553/1, EP/N014588/1, EP/T017961/1, the Wellcome Innovator Award RG98755, the Leverhulme Trust project Unveiling the invisible, the European Union Horizon 2020 research and innovation programme under the Marie Skodowska-Curie grant agreement No. 777826 NoMADS, the Cantab Capital Institute for the Mathematics of Information and the Alan Turing Institute. CE acknowledges support from the Wellcome Innovator Award RG98755 for part of the work that was done at Cambridge.

Chapter 5

Diffusion models encode the intrinsic dimension of data manifolds

In this work, we provide a mathematical proof that diffusion models encode data manifolds by approximating their normal bundles. Based on this observation we propose a novel method for extracting the intrinsic dimension of the data manifold from a trained diffusion model. Our insights are based on the fact that a diffusion model approximates the score function i.e. the gradient of the log density of a noise-corrupted version of the target distribution for varying levels of corruption. We prove that as the level of corruption decreases, the score function points towards the manifold, as this direction becomes the direction of maximal likelihood increase. Therefore, at low noise levels, the diffusion model provides us with an approximation of the manifold's normal bundle, allowing for an estimation of the manifold's intrinsic dimension. To the best of our knowledge our method is the first estimator of intrinsic dimension based on diffusion models and it outperforms well established estimators in controlled experiments on both Euclidean and image data. The code is available at https://github.com/GBATZOLIS/ID-diff.

5.1 Introduction

Many modern real-world datasets contain a large number of variables, often exceeding the number of observations. This poses a major challenge in modelling them, due to the *curse of dimensionality*. Despite this complexity, due to the numerous relationships and symmetries among variables, even high-dimensional data often concentrates around a lower-dimensional manifold, a concept known as the *manifold hypothesis* [53]. The dimension of this manifold

is called *intrinsic dimension* (ID), while the high-dimensional space in which the data resides is known as the *ambient space*, with its dimensionality called the *ambient dimension*.

The manifold hypothesis has guided the development of modern high-dimensional data modelling techniques, such as Variational auto-encoders (VAEs) [104], Generative Adversarial Networks (GANs) [63] and M-flows [22].

The estimation of ID holds significant importance in the machine learning community due to its applicability in both theoretical and practical problems [26]. From a theoretical perspective, the ID is essential as it directly affects the convergence rates of fundamental statistical quantities [212]. The higher the ID, the more data is needed for a model to generalize well beyond the training set [26, 154], hence knowing the ID has numerous implications for the generalization and data efficiency of machine learning models [99, 106]. From practical point of view, ID is crucial for a wide range of dimensionality reduction methods [26]. Additionally, understanding the data's ID can help in fine-tuning the latent dimension of models such as GANs, VAEs or M-flows.

In recent years, diffusion models [178, 78] emerged as a new class of deep generative models capable of capturing complex high-dimensional distributions without relying on the notion of data manifold or prior knowledge of the data's ID. Our research reveals that diffusion models encode data manifolds via their normal bundle. Intriguingly, we find that while diffusion models do not *explicitly* rely on the ID, these models estimate it *implicitly*.

As discussed in [183, 78], diffusion models perform score matching [85] and, therefore, contain the information about the gradient of the log-density of the data distribution. We prove that near the data manifold, the gradient of the log-density is orthogonal to the manifold itself. This key observation serves as a tool for deducing the manifold's dimension.

In our study, we investigate three categories of ID estimators: traditional statistical methods (such as PCA and Nearest Neighbor based approaches), normalizing flow-based methods, and our innovative diffusion-based approach. We evaluate the performance of ID estimators on synthetic Euclidean and image datasets, where the dimension of the data manifold is known *a priori*. Moreover, we apply ID estimators to the MNIST dataset [114] (where the ID is unknown), and compare the estimated IDs with the reconstruction error of auto-encoders trained with different latent dimensions.

Our findings indicate that in datasets of high ID, methods that exploit the inductive biases of neural networks are the most effective. Our proposed method stands out by yielding the best results. This success is attributed to utilizing diffusion models, which offer enhanced training stability and avoid the architectural limitations associated with normalizing flows [16].

To summarize our contributions are as follows:

- We elucidate a geometric connection between diffusion models and data manifolds, by proving that a diffusion model encodes the data manifold by approximating its normal bundle.
- Based on this observation we propose a novel method for extracting the ID of the data manifold from a trained diffusion model.
- We perform an extensive evaluation of our novel method as well as several prominent existing methods for ID estimation on a wide range of datasets.

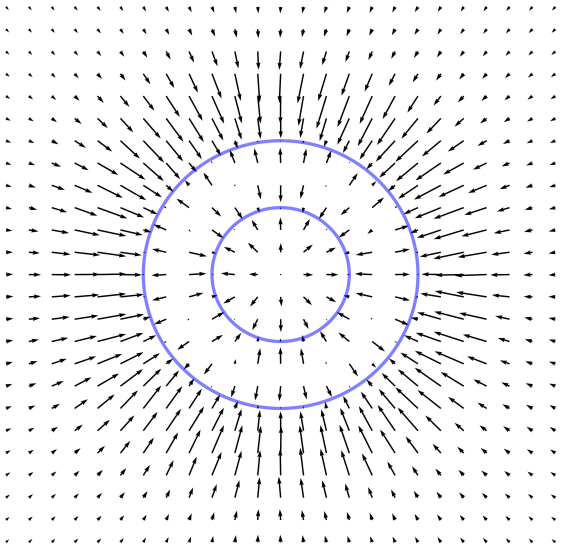


Fig. 5.1 The data manifold (in blue) and the neural approximation of the score field $\nabla_{\mathbf{x}} \ln p_{t_0}(\mathbf{x})$ obtained from a diffusion model. Near the manifold the score field is perpendicular to the manifold surface.

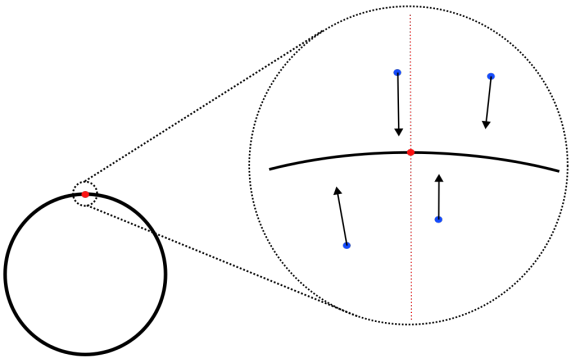


Fig. 5.2 To estimate the manifold's dimension at point $\mathbf{x}_0$ (red dot), we sample K nearby points $\mathbf{x}_\varepsilon^{(i)}$ (blue dots) and use the trained diffusion model to evaluate the score function $s_\theta(\mathbf{x}_\varepsilon^{(i)}, \varepsilon)$ at these perturbed points. We assemble these vectors into a matrix and perform Singular Value Decomposition (SVD). The number of (almost) zero singular values reveals the manifold's dimension.

5.2 Related Work

The relationship between diffusion models and manifold hypothesis has been explored in several recent works. In [152] author examines theoretical conditions under which diffusion models produce samples from the underlying data manifold. In [138] and [30] authors

analyze approximation and generalization abilities of diffusion models under manifold hypothesis. They establish that the data efficiency of training a diffusion model depends on the intrinsic dimension of the data manifold rather than the ambient dimension. This further motivates the importance of ID estimation.

The problem of estimating the intrinsic dimensionality has been widely studied. The two main lines of research are PCA based and nearest neighbour based approaches. In an early work [56] the authors suggest an approach based on using local Karhunen–Loève expansion. In following years many PCA based approaches have been developed. Most notably, in [131] the author suggests an intrinsic dimensionality estimator based on the probabilistic PCA (PPCA) framework [19]. In [52] a local PCA method has been suggested. In [151] authors suggested an estimator based on nearest neighbour information. In [117] authors introduce a maximum likelihood (MLE) procedure based on the distance to m nearest neighbours. Their method has been further improved in the work of [69]. The MLE method has been recently applied by [154] in the estimation of the intrinsic dimensionality of modern image datasets such as MNIST [114], CIFAR [107] and ImageNet [39]. Other works explored geometric approaches using fractal-based methods [25] or packing numbers [96]. We refer to [26] for a comprehensive survey of statistical approaches to ID estimation.

The aforementioned ID estimators do not leverage the inductive bias introduced by modern neural network architectures, which is a crucial reason for the success of modern deep leaning systems [64]. Therefore, their statistical efficiency may be insufficient to deal with datasets of high ID. This limitation has been observed in [26, 80] and is confirmed by our experimental findings.

The limitations of traditional statistical estimators have recently led to the development of deep learning-based intrinsic dimensionality (ID) estimators such as LIDL [197] and ID-NF [80]. These methods, which extract ID from trained normalizing flows, outperform their statistical counterparts by leveraging the inductive biases inherent in modern deep neural network architectures. Despite their advantages, they face challenges as they rely on normalizing flows which are invertible neural networks. Normalizing flows are subject to a trade-off between stability and expressivity, as discussed in [16],[89], [37],[109]. Expressive normalizing flow architectures often face stability problems during training or evaluation, risking the reliability of post-training ID estimation due to potential loss of numerical invertibility. On the other hand, Lipschitz constrained normalizing flow architectures, while more stable, tend to be less expressive, potentially limiting their capacity to accurately estimate the ID of complex, high-dimensional data manifolds.

In contrast, our proposed methodology, which utilizes diffusion models, effectively sidesteps these expressivity limitations as diffusion models do not suffer from similar stability

issues. Our method allows for more effective leveraging of the powerful inductive biases of modern deep neural network architectures, resulting in a more reliable and robust estimation of the ID of complex, high-dimensional data manifolds. This claim is further substantiated by our experimental findings, which show that our proposed diffusion-based estimation method accurates estimes the ID in challenging data manifolds where normalizing flow-based methods fail due to reduced expressivity.

5.3 Proposed Method for Estimation of Intrinsic Dimension

Algorithm 1 Estimate the Intrinsic Dimension at $\mathbf{x}_0$

Input: s_θ - trained diffusion model, $\mathbf{x}_0$ - data point, K - number of samples
Sample $\mathbf{x}_0 \sim p_0(\mathbf{x})$ from the data set
$d \leftarrow \dim(\mathbf{x}_0)$
$S \leftarrow$ empty matrix
for $i = 1, \ldots, K$ **do**
    Sample $\mathbf{x}_{t_0}^{(i)} \sim \mathcal{N}(\mathbf{x}_0, \sigma^2 I)$
    Append $s_\theta(\mathbf{x}_{t_0}^{(i)}, t_0)$ as a new row in S
end for
$(s_i)_{i=1}^d, (\mathbf{v}_i)_{i=1}^d, (\mathbf{w}_i)_{i=1}^d \leftarrow \mathrm{SVD}(S)$
$\hat{k}(\mathbf{x}_0) \leftarrow d - \arg\max_{i=1,\ldots,d-1}(s_i - s_{i+1})$
Output: $\hat{k}(\mathbf{x}_0)$

In [183] score-based [85] and diffusion-based [178, 78] generative models were unified into a single continuous-time score-based framework. The diffusion process is represented by a stochastic differential equation (SDE), which perturbs data distribution p_0 resulting in a series of progressively noise-corrupted distributions p_t. Diffusion models are trained to approximate the score function $\nabla_{\mathbf{x}_t} \ln p_t(\mathbf{x}_t)$ with a neural network $s_\theta(\mathbf{x}_t, t)$. Once the score function is approximated, the diffusion SDE can be reversed to generate samples from p_0. Additional details on training and sampling from diffusion models are described in Appendix C.1.

Consider a dataset $D = \{\mathbf{x}^{(i)}\}_{i=0}^N \sim p_0(\mathbf{x})$ which consists of N independent d-dimensional vectors $\mathbf{x}^{(i)} \in \mathbb{R}^d$ drawn from distribution $p_0(\mathbf{x})$. The distribution $p_0(\mathbf{x})$ is supported on a k-dimensional manifold $\mathscr{M}$, which is embedded in a space of ambient dimension d. Our goal is to infer the dimension k of the manifold $\mathscr{M}$ from D.

We perturb the data according to the variance exploding SDE $d\mathbf{x}_t = g(t)d\mathbf{w}_t$ [183] and train a neural network $s_\theta(\mathbf{x}_t, t)$ to approximate the score function of the noise perturbed target distribution, i.e. $\nabla_{\mathbf{x}_t} \ln p_t(\mathbf{x}_t)$ for a range of levels of perturbation indexed by diffusion time t.

We train the model using the weighted denoising score matching objective with likelihood weighting, see [181]. More details about the training of the diffusion model can be found in Appendix C.2.

Consider a datapoint $\mathbf{x}_0$ on a manifold $\mathscr{M}$ and its perturbation into the ambient space $\mathbf{x}_{t_0}$ obtained from the transition kernel $p_{t_0}(\mathbf{x}_{t_0}|\mathbf{x}_0) = \mathscr{N}(\mathbf{x}_{t_0}|\mathbf{x}_0, \sigma_{t_0}^2\mathbf{I})$[1] of the forward process at a small time t_0. As shown in the next section, at $\mathbf{x}_{t_0}$ the score vector $s_\theta(\mathbf{x}_{t_0}, t_0)$ will point towards its orthogonal projection onto $\mathscr{M}$, making it almost orthogonal to $T_{\mathbf{x}_0}\mathscr{M}$ (the tangent space at $\mathbf{x}_0$). This means that the projection of the score vector onto the normal space $\mathscr{N}_{\mathbf{x}_0}\mathscr{M}$ will be significantly larger than its projection onto the tangent space $T_{\mathbf{x}_0}\mathscr{M}$. Therefore, with enough samples, the spectrum obtained from the singular value decomposition of $S = [s_\theta(\mathbf{x}_{t_0}^{(1)}, t_0), ..., s_\theta(\mathbf{x}_t^{(K)}, t_0)]$ will have N_d large singular values and T_d very small singular values, where $N_d = \dim(\mathscr{N}_{\mathbf{x}_0}\mathscr{M})$ and $T_d = \dim(T_{\mathbf{x}_0}\mathscr{M})$. This will be the case because the projection of the score vector at every perturbed point considered is much larger on the normal space than on the tangent space. In our method, we sample $K = 4d$ diffused points at time $t_0 = \varepsilon$ and calculate the SVD of S. The number of vanishing singular values is the estimate of the intrinsic dimension $\hat{k}(\mathbf{x}_0)$.

The resulting spectrum shows a significant drop exactly or very close to the dimension of the normal space. The remaining non-zero, but much smaller singular values correspond to the tangential component of the score vector. This behaviour is expected as the score vector will unavoidably have a very small tangential component, for reasons explained in the following sections. The choice of the cut-off point $\hat{k}(\mathbf{x}_0)$ is usually very clear visually, but can be automated by choosing the point of largest drop in the spectrum:

$$\hat{k}(\mathbf{x}_0) = d - \underset{i=1,..,d-1}{\arg\max}\, s_i - s_{i+1}$$

When selecting $\mathbf{x}_0$, we ideally want a point with high score approximation quality, minimal tangential component, and low manifold curvature. However, since these factors are uncontrollable, we randomly choose multiple $\mathbf{x}_0^{(j)}$ values and plot a spectrum for each. For simple distributions, the score spectra look similar, with drops at accurate values. For more complex distributions, the drop location varies with $\mathbf{x}_0^{(j)}$ choice. We find that the maximum estimated $\hat{k}$ gives the best estimate. Theoretical understanding of the method supports this, as discussed in later sections.

[1] The transition kernels of the variance exploding SDE have this structure, where σ_t is an increasing function determined by $g(t)$ with $\sigma_t \xrightarrow[t\to 0]{} 0$.

5.4 Theoretical Analysis

Here, we provide a theoretical justification for our approach. We demonstrate that, given a collection of points $\mathbf{x}_i \in \mathbb{R}^d$ sufficiently close[2] to the manifold $\mathscr{M}$ with orthogonal projection $\pi(\mathbf{x}_i) \in \mathscr{M}$, the space spanned by the score vectors $\nabla_{\mathbf{x}} \ln p_t(\mathbf{x}_i)$ converges to the normal space at $\pi(\mathbf{x}_i)$ in the small t limit. To build intuition, consider a uniform data density on the manifold surface $\mathscr{M}$. The gradient along this density is zero, indicating that for $\mathbf{x}$ close to $\mathscr{M}$ tangential components of the score $\nabla_{\mathbf{x}} \ln p_t(\mathbf{x})$ will also be approximately zero and the score will be mostly contained in the normal bundle $\mathscr{N}\mathscr{M}$. If the density is non-uniform on the manifold surface however, the score will have a tangential component. Fortunately, for sufficiently small t the change in log-density from moving orthogonally towards the manifold dominates the change from moving tangentially alongside the manifold. This results in the tangential component becoming negligible, and the score still being approximately contained in $\mathscr{N}\mathscr{M}$.

Specifically, we show in the following theorem that for any point $\mathbf{x}$ sufficiently close to the data manifold and $t \to 0$, the score $\nabla_{\mathbf{x}} \ln p_t(\mathbf{x})$ points directly at the orthogonal projection $\pi(\mathbf{x})$[3].

Theorem 5.4.1. *Suppose that the support of the data distribution P_0 is contained in a compact embedded sub-manifold $\mathscr{M} \subseteq \mathbb{R}^d$ and let P_t be the distribution of samples from P_0 diffused for time t. Then, under mild assumptions (see Appendix C.4), for any point $\mathbf{x} \in \mathbb{R}^d$ sufficiently close to $\mathscr{M}$ with orthogonal projection on $\mathscr{M}$ given by $\pi(\mathbf{x})$, and $\mathbf{n} = (\pi(\mathbf{x}) - \mathbf{x}) / \|\pi(\mathbf{x}) - \mathbf{x}\|$, we have:*

$$S_{\cos}(\mathbf{n}, \nabla_{\boldsymbol{x}} \ln p_t(\mathbf{x})) \xrightarrow[t \to 0]{} 1$$

where $S_{\cos}$ denotes the cosine similarity, defined as $S_{\cos}(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \|\mathbf{b}\|}$. In other words, for sufficiently small t the score $\nabla_{\boldsymbol{x}} \ln p_t(\mathbf{x})$ points directly at the projection of $\mathbf{x}$ on the manifold.

This theorem leads to the conclusion that this score is contained within the normal space of the manifold, as we show with the following corollary:

Corollary 5.4.2. *The ratio of the projection of the score $\nabla_{\boldsymbol{x}} \ln p_t(\mathbf{x})$ on the tangent space of the data manifold $T_{\pi(\mathbf{x})}\mathscr{M}$ to the projection on the normal space $\mathscr{N}_{\pi(\mathbf{x})}\mathscr{M}$ approaches zero*

[2] Within the tubular neighbourhood of $\mathscr{M}$ to the manifold $\mathscr{M}$. See Appendix C.4.

[3] Every compact embedded sub-manifold $\mathscr{M}$ has a tubular neighbourhood, and every point $\mathbf{x}$ in the tubular neighbourhood of the manifold has a unique orthogonal projection $\pi(\mathbf{x})$ onto the manifold. See Appendix C.4 for more details.

as t approaches zero, i.e.

$$\frac{\|\mathbf{T}\nabla_{\boldsymbol{x}} \ln p_t(\mathbf{x})\|}{\|\mathbf{N}\nabla_x \ln p_t(\mathbf{x})\|} \to 0, \text{ as } t \to 0.$$

where $\mathbf{N}$ *and* $\mathbf{T}$ *are projection matrices on* $\mathcal{N}_{\pi(\mathbf{x})}\mathcal{M}$ *and* $T_{\pi(\mathbf{x})}\mathcal{M}$ *respectively. Therefore for sufficiently small* t *the score* $\nabla_{\mathbf{x}} \ln p_t(\mathbf{x})$ *is (effectively) contained in the normal space* $\mathcal{N}_{\pi(\mathbf{x})}\mathcal{M}$.

Proof. The full proof of the theorem and corollary can be found in the Appendix C.4.

In practice we choose small $t > 0$, and for each chosen $\mathbf{x}_0 \in \mathcal{M}$ we sample points $\mathbf{x}_t^{(i)}$ around it from $p_t(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t|\mathbf{x}_0, \sigma_t^2\mathbf{I})$. With over 99% probability $\mathbf{x}_t^{(i)} \in B(\mathbf{x}_0, 3\sigma_t)$, and so as we decrease t most of our $\mathbf{x}_t$ will become very close to $\mathbf{x}_0$. For $B(\mathbf{x}_0, 3\sigma_t)$ sufficiently small, the effect of curvature of $\mathcal{M}$ becomes negligible inside $B(\mathbf{x}_0, 3\sigma_t)$, and so the normal spaces $\mathcal{N}_{\pi(\mathbf{x}_t^{(i)})}\mathcal{M}$ will all be approximately equal to $\mathcal{N}_{\mathbf{x}_0}\mathcal{M}$. Under this assumption, we can outline the practical implications of these theoretical results. Assuming $t > 0$ small and a trained score approximation $s_\theta(\mathbf{x}, t) \approx \nabla_{\mathbf{x}} \ln p_t(\mathbf{x})$, the score matrix $S = [s_\theta(\mathbf{x}_t^{(1)}, t), ..., s_\theta(\mathbf{x}_t^{(4d)}, t)]$ has the following properties:

1. The columns of S are approximately contained in the normal space $\mathcal{N}_{\mathbf{x}_0}\mathcal{M}$
2. The columns of S approximately span the normal space $\mathcal{N}_{\mathbf{x}_0}\mathcal{M}$
3. The singular values of S corresponding to singular vectors in the normal space are large relative to those corresponding to tangent singular vectors

The first point is a direct consequence of Corollary 5.4.2. For the second point, denote $n_{\mathbf{x}} := \frac{\pi(\mathbf{x})-\mathbf{x}}{\|\pi(\mathbf{x})-\mathbf{x}\|}$, and assume that t is sufficiently small such that $\mathcal{N}_{\pi(\mathbf{x}_t^{(i)})}\mathcal{M} \approx \mathcal{N}_{\mathbf{x}_0}\mathcal{M}$. Then locally, the vectors $\left\{n_{\mathbf{x}_t^{(1)}}, \ldots, n_{\mathbf{x}_t^{(K)}}\right\}$ are independent Gaussian perturbations from a linear subspace. With probability one this set contains $N_d = \dim(\mathcal{N}_{\mathbf{x}_0}\mathcal{M})$ linearly independent vectors spanning $\mathcal{N}_{\mathbf{x}_0}\mathcal{M}$, so by Theorem 5.4.1 the score vectors $\left\{\nabla_{\mathbf{x}} \ln p_t\left(\mathbf{x}_t^{(1)}\right), \ldots, \nabla_{\mathbf{x}} \ln p_t\left(\mathbf{x}_t^{(K)}\right)\right\}$ must therefore also span $\mathcal{N}_{\mathbf{x}_0}\mathcal{M}$. For the third point note that if the columns of S span $\mathcal{N}_{\mathbf{x}_0}\mathcal{M}$, then S has rank N_d, and its SVD yields singular values $s_i > 0$ for $i \leq N_d$ corresponding to singular vectors in $\mathcal{N}_{\mathbf{x}_0}\mathcal{M}$, and $s_j = 0$ for $j > N_d$. In practice we fix some small $t > 0$ and so small components of $T_{\mathbf{x}_0}\mathcal{M}$ are introduced to the score by factors such as non-uniform distribution of data samples on $\mathcal{M}$, therefore in applications the SVD of S yields small singular values $s_j > 0$ for $j > N_d$, however we still observe that $s_i \gg s_j$ where $i \leq N_d$.

To formalize the idea of "approximately spanning the normal space" and $\mathcal{N}_{\pi(\mathbf{x}_t^{(i)})}\mathcal{M} \approx \mathcal{N}_{\mathbf{x}_0}\mathcal{M}$, we use the concept of cosine similarity and the angle between subspaces. Cosine

similarity measures the cosine of the angle between two vectors, which helps quantify how closely two vectors (or subspaces) align.

Let $v_1, \ldots, v_k$ be vectors sampled from the normal space $\mathcal{N}_{\mathbf{x}_0}\mathcal{M}$. We define the cosine similarity between two vectors v_i and v_j as:

$$\cos(\theta_{ij}) = \frac{v_i \cdot v_j}{\|v_i\|\|v_j\|},$$

where θ_{ij} is the angle between v_i and v_j.

For subspaces $\mathcal{N}_{\mathbf{x}_0}\mathcal{M}$ and $\mathcal{N}_{\pi(\mathbf{x}_t^{(i)})}\mathcal{M}$, we consider the principal angles $\theta_1, \ldots, \theta_k$ between them. If the cosine of these angles is close to 1, the subspaces are well-aligned:

$$\cos(\theta_k) \approx 1 \quad \text{implies} \quad \mathcal{N}_{\pi(\mathbf{x}_t^{(i)})}\mathcal{M} \approx \mathcal{N}_{\mathbf{x}_0}\mathcal{M}.$$

Thus, by evaluating the cosine similarities and the angles between the subspaces, we can quantify the approximation quality of spanning the normal space. This formalization supports the intuitive claims by providing a measurable criterion for the alignment of normal spaces.

5.5 Limitations

In section 5.4, we established that given a perfect score approximation for sufficiently low t our method produces the correct estimation of the dimension. However, in practice, our method may encounter two types of errors: *approximation error* and *geometric error*. The approximation error arises as a result of having an imperfect score approximation $s_\theta(\mathbf{x}, t) \approx \nabla_{\mathbf{x}} \ln p_t(\mathbf{x})$.

Geometric error arises if the selected sampling time t isn't sufficiently small, potentially impacting our method's accuracy for two reasons. Firstly, it may result in an increased tangential component of the score vector. Secondly, if $\mathbf{x}_t^{(i)}$ lies too distant from $\mathcal{M}$, the manifold's curvature may create a difference between normal spaces $\mathcal{N}_{\pi(\mathbf{x}_t^{(i)})}\mathcal{M}$ across varying i.

We empirically assess our method's robustness to approximation error and find it robust to minor inaccuracies in score approximation. Additionally, we analyze our method's sensitivity to p_0 non-uniformity, which could induce a minor tangential score component for $t > 0$. We discover that using the maximum $\hat{k}(\mathbf{x}_0^{(j)})$ allows our method to accommodate varying levels of non-uniformity over the manifold surface, displaying superior robustness compared to other non-linear estimators without the need for reducing t. Details are in Appendix C.7

We note that Theorem 5.4.1 assumes that the data distribution's support is exclusively within a manifold. Therefore, we empirically investigate the method's applicability when data is concentrated around, but not entirely within, a manifold. We discover that for a k-sphere, our method remains reliable as long as the data is closely concentrated around the manifold. Details are available in Appendix C.7.

5.6 Experiments

We examine the effectiveness of our method on a multitude of manifold datasets, each embedded in a high-dimensional ambient space. The datasets fall into two categories: *Euclidean datasets* consisting of points from manifolds embedded in a high-dimensional Euclidean space and *image datasets* consisting of synthetic manifolds of images. In each case we know the intrinsic dimension of the manifold *a priori*. Additionally, we apply our method to the MNIST dataset, where the true intrinsic dimension is unknown. We assess its performance via comparison with the reconstruction error of auto-encoders with various latent dimensions. For each dataset we train a diffusion model, and then apply our method to estimate the intrinsic dimension of the data manifold. Details on hyperparameters and architectures used in our experiments can be found in Appendix C.2.

We compare our method against established approaches to intrinsic dimensionality estimation: the nearest neighbour based maximum likelihood estimator (MLE) [117], [69], Local PCA [52] and Probabilistic PCA (PPCA) [131] [19]. Additionally, we compare our method against ID-NF [80], which is the best performing method for extracting the ID from pretrained normalizing flows. The details about the implementation of the benchmarks are in the Appendix C.3.

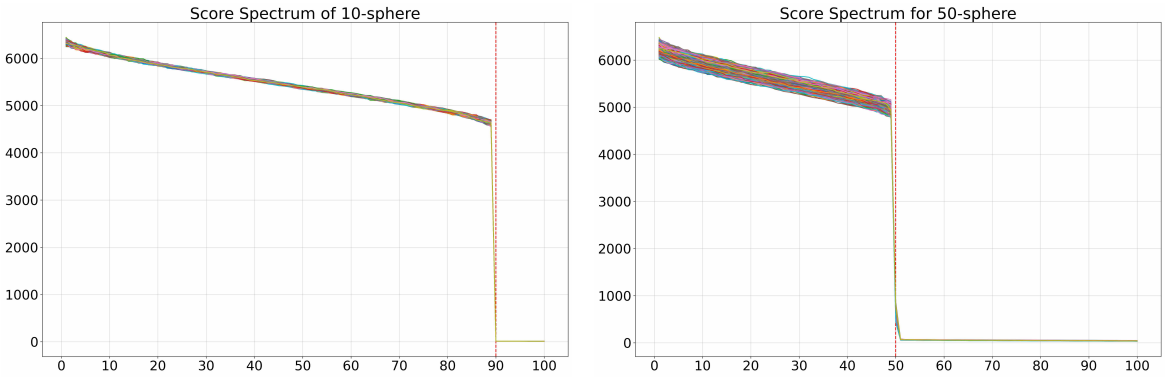
Our method consistently yields the best estimate or close to the best estimate among considered approaches. In the following subsections we present a detailed discussion of each experiment. The results are summarised in Table 5.2.

5.6.1 Experiments on Euclidean datasets

Embedded k-spheres: We examined our method on k dimensional spheres embedded in a $d = 100$ dimensional ambient space via a random isometric embedding[4]. We consider two cases $k_1 = 10$ and $k_2 = 50$. The spectra of resulting score matrices are presented in Figure

[4]To obtain a random isometric embedding we first generate the k sphere in a $k+1$ dimensional small ambient space. Then we sample a random $d \times (k+1)$ Gaussian matrix A. We perform a QR decomposition $A = QR$. Finally we use the $d \times (k+1)$ isometry matrix Q to embed the small $k+1$ dimensional space containing our manifold in the large d dimensional ambient space.

5.3. Our method gives estimates of $\hat{k}_1 = 11$ and $\hat{k}_2 = 51$, which are very close to the true intrinsic dimensionality of the manifolds.



(a) $k = 10$ (b) $k = 50$

Fig. 5.3 Singular values for the scores of k-sphere for $k = 10, 50$. In both cases around k singular values almost vanish, clearly indicating the dimensionality of the manifold. Each line shows a score spectrum at different $\mathbf{x}_0^{(j)}$.

Spaghetti line: The intrinsic dimensionality of k-spheres could be well approximated by a linear dimensionality detection methods such as [131]. This is because these manifolds are contained in a low dimensional *linear* subspace. In order to showcase the advantage of the *non-linear* nature of our method we consider a *spaghetti line* manifold. That is a curve $\tau \mapsto (\sin(\tau), \sin(2\tau), ..., \sin(100\tau))$ in a 100 dimensional ambient space, which is not contained in any low dimensional linear subspace (cf. Figure C.4 in Appendix C.6.1). As expected, the linear method [131] greatly overstates the intrinsic dimension with a result of $\hat{k}_{\text{PPCA}} = 98$. Yet, our approach utilizes the non-linear knowledge from the diffusion model to accurately predict an intrinsic dimensionality of one. The score spectrum is presented in Figure C.5 in Appendix C.6.1.

Union of k-spheres: Due to the local nature of our method, we are able to generate an estimate $\hat{k}(\mathbf{x}_0)$ of the intrinsic dimension around a given point $\mathbf{x}_0$. This allows us to apply our approach to a union of manifolds and identify the dimension of each component. We illustrate this feature with the following experiment. We embed two spheres of different radii and dimensions in a 100 dimensional ambient space. First sphere has dimension $k_1 = 10$ and radius $r_1 = 1$ and the second sphere has $k_2 = 30$ and radius $r_2 = 0.25$[5]. We apply our method to this data using multiple $\mathbf{x}_0^{(i)}$ randomly sampled from the dataset. We observe that our method produces a spectrum with two visible, separated drops. This indicates that the data comes from the union of manifolds of different dimensions. The resulting estimates are

[5] One can intuitively think of this manifold as a high-dimensional analog of a planet with a ring around it.

$\hat{k}_1 = 10$ and $\hat{k}_2 = 31$ depending on the chosen $\mathbf{x}_0^{(j)}$. The score spectra and the histogram of estimated dimensions are presented in Figures C.6 and C.7 in Appendix C.6.1.

5.6.2 Experiments on image datasets

Synthetic image manifolds: In this experiment, we investigated our method's ability to infer the dimension of synthetic image manifolds with known dimension. We crafted two synthetic image manifolds with controllable intrinsic dimension k: the "k squares images manifold" and the "k Gaussian blobs images manifold". The construction of these manifolds is detailed in Appendix C.5. We evaluated our method on k =10, 20, and 100 dimensional manifolds for both types, with the score spectra and histograms of estimated dimensions for numerous data points displayed in Figures C.8 and C.9 in Appendix C.6.2.

On the squares image manifold, our method, ID-NF and PPCA consistently yielded accurate dimension estimates. PPCA's success on this dataset was anticipated since the manifold resides within a k-dimensional linear subspace.

On the more complex Gaussian blobs image manifold, our method stood out as the sole technique to consistently deliver accurate dimension estimates. The accuracy of our method was not compromised by the manifold's increased complexity, unlike other methods. However, the estimation for the 100-dimensional manifold introduced some uncertainty, as indicated by a more leveled histogram and a less abrupt spectrum collapse (c.f. Figure C.9). This is attributed to the manifold's increased complexity and the inherent challenges in optimization, resulting in greater geometric and approximation errors.

MNIST: In our study, we additionally applied the proposed technique to estimate the intrinsic dimension of the well-known MNIST dataset - an image dataset with an as-of-yet-undetermined intrinsic dimension. Our findings suggest that there exists a variation in the intrinsic dimensions across different digits. For instance, the digit '1' yielded an estimated dimension of 66, whereas the digit '9' exhibited a significantly higher estimated dimension of 152. This discrepancy can be attributed to the increased geometric complexity inherent to the digit '9'. Figure 5.5 elucidates these observations by displaying the score spectra which yielded the maximum estimated dimensions for each digit. We present the estimated dimension for each digit in Table 5.1 and the complete set of spectra for each digit in the Appendix C.6.3.

We validate our estimates by comparing them with the reconstruction error of auto-encoders trained with different latent dimensions. As demonstrated in Figure 4, the ID estimate of our method is in close agreement with that of the ID-NF method, and both correlate with the point of diminishing returns on the reconstruction loss curve. This point marks a plateau in the effectiveness of additional latent dimensions to significantly reduce

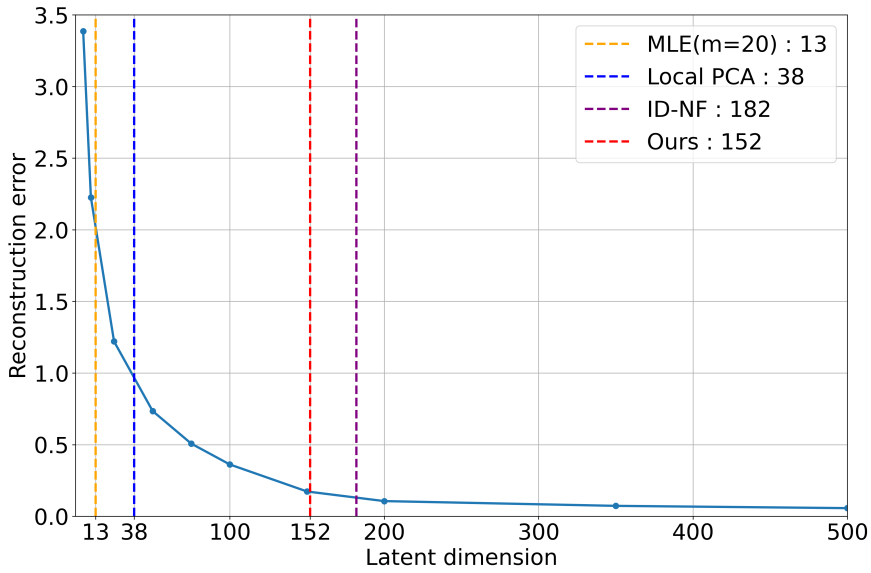


Fig. 5.4 Auto-encoder reconstruction error on MNIST for different latent space dimensions. Vertical lines mark different estimations of intrinsic dimension.

reconstruction error, further suggesting this point as the dataset's intrinsic dimension. In constrast, estimates produced by MLE and Local PCA are significantly lower, corresponding to regions of the curve where the reconstruction loss is still steeply decreasing. This suggests these methods underestimate the manifold dimension. These findings call for a careful interpretation of the intrinsic dimension estimates of popular machine learning datasets provided by [154], as they rely on the MLE method, which we have found to consistently underestimate manifold dimensions. On the other hand, PPCA notably overestimated the dimension, with $\hat{k}_{\text{PPCA}} = 706$.

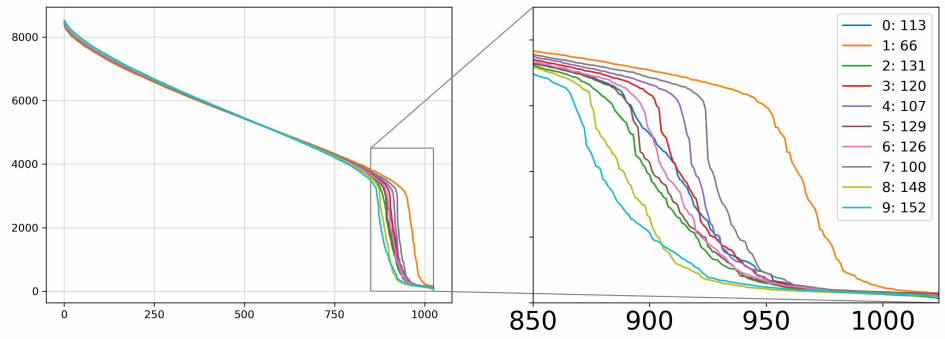


Fig. 5.5 MNIST score spectra that yielded the highest estimated dimension for each digit

0	1	2	3	4	5	6	7	8	9
113	66	131	120	107	129	126	100	148	152

Table 5.1 Estimated intrinsic dimension for each MNIST digit

	Ground Truth	Ours	ID-NF	MLE (m=5)	MLE (m=20)	Local PCA	PPCA
Euclidean Data Manifolds							
10-sphere	10	11	11	9.61	9.46	11	11
50-sphere	50	51	51	35.52	34.04	51	51
Spaghetti line	1	1	1	1.01	1.00	32	98
Image Manifolds							
Squares							
$k = 10$	10	11	9.7	8.48	8.17	10	10
$k = 20$	20	22	19.5	14.96	14.36	20	20
$k = 100$	100	100	94.2	37.69	34.42	78	99
Gaussian blobs							
$k = 10$	10	12	9.8	8.88	8.67	10	136
$k = 20$	20	21	17.8	16.34	15.75	20	264
$k = 100$	100	98	56.3	39.66	35.31	18	985
MNIST	N/A	152	182	14.12	13.27	38	706

Table 5.2 Comparison of dimensionality detection methods on various data manifolds.

5.7 Conclusions and further directions

In this work, we proved theoretically and confirmed experimentally that diffusion models can infer the intrinsic dimension from the data. We introduced an approach that estimates the intrinsic dimension of the data manifold from a pre-trained diffusion model. This approach capitalizes on the observation that, the diffusion model evaluated at sufficiently small diffusion time approximates the normal bundle of the data manifold. Our work offers a twofold contribution: it highlights that diffusion model detects the lower dimensional structure of data and provides a rigorous method for intrinsic dimension estimation.

We conducted a rigorous comparison of three types of ID estimators: traditional statistical methods, normalizing flow-based techniques, and our diffusion-based approach. Our findings consistently show that for high-ID datasets, methods leveraging neural networks' inductive biases are superior. Notably, our diffusion-based method emerges as the most effective, owing to its enhanced training stability and freedom from the architectural constraints of normalizing flows.

Furthermore, our research introduces new estimates for the MNIST's dimensionality, demonstrating strong alignment with the predictions of an auto-encoder trained across a range of latent dimensions.

Our work opens new paths for understanding and estimating intrinsic data dimension, with potential implications across the field of machine learning. Future research should explore this method's applicability to other data types and its potential across various domains.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

Acknowledgements

JS aknowledges support from the Cantab Capital Institute for the Mathematics of Information and Aviva. GB acknowledges support from GSK. TD acknowledges support from the EPSRC programme grant in 'The Mathematics of Deep Learning', under the project EP/L015684/1. CBS acknowledges support from the Philip Leverhulme Prize, the Royal Society Wolfson Fellowship, the EPSRC advanced career fellowship EP/V029428/1, the EPSRC programme grant EP/V026259/1, and the EPSRC grants EP/S026045/1 and EP/T003553/1, EP/N014588/1, EP/T017961/1, the Wellcome Innovator Awards 215733/Z/19/Z and 221633/Z/20/Z, the European Union Horizon 2020 research and innovation programme under the Marie Skodowska-Curie grant agreement No. 777826 NoMADS, the Cantab Capital Institute for the Mathematics of Information and the Alan Turing Institute. This research was supported by the NIHR Cambridge Biomedical Research Centre (NIHR203312). The views expressed are those of the author(s) and not necessarily those of the NIHR or the Department of Health and Social Care.

Chapter 6

Variational Diffusion Auto-encoder

As a widely recognized approach to deep generative modeling, Variational Auto-Encoders (VAEs) still face challenges with the quality of generated images, often presenting noticeable blurriness. This issue stems from the unrealistic assumption that approximates the conditional data distribution, $p(\mathbf{x}|\mathbf{z})$, as an isotropic Gaussian. In this paper, we propose a novel solution to address these issues. We illustrate how one can extract a latent space from a pre-existing diffusion model by optimizing an encoder to maximize the marginal data log-likelihood. Furthermore, we demonstrate that a decoder can be analytically derived post encoder-training, employing the Bayes rule for scores. This leads to a VAE-esque deep latent variable model, which discards the need for Gaussian assumptions on $p(\mathbf{x}|\mathbf{z})$ or the training of a separate decoder network. Our method, which capitalizes on the strengths of pre-trained diffusion models and equips them with latent spaces, results in a significant enhancement to the performance of VAEs.

6.1 Introduction

Variational Autoencoders (VAEs) [104] have proven to be a powerful tool for unsupervised learning, allowing for the efficient modeling and generation of complex data distributions. However, VAEs have important limitations, including difficulty in capturing the underlying structure of high-dimensional data and generating blurry images [223]. These problems emerge due to an unrealistic modeling assumption that the conditional data distribution $p(\mathbf{x}|\mathbf{z})$ can be approximated as a Gaussian distribution. Moreover, instead of sampling from $p(\mathbf{x}|\mathbf{z})$ the model simply outputs the mean of the distribution, which results in an undesirable smoothing effect. In this work we propose to relax this limiting assumption by modeling $p(\mathbf{x}|\mathbf{z})$ in a flexible way by leveraging the capabilities of diffusion models. We show that an

encoder network modeling $p(\mathbf{z}|\mathbf{x})$ can be easily combined with an unconditional diffusion model trained on $p(\mathbf{x})$ to yield a model for $p(\mathbf{x}|\mathbf{z})$.

Diffusion models [178, 78] have recently emerged as a promising technique for generative modeling, which uses the time reversal of a diffusion process, to estimate the data distribution $p(\mathbf{x})$. Diffusion models proved to be incredibly successful in capturing complex high-dimensional distributions achieving state-of-the-art performance in many tasks such as image synthesis [42] and audio generation [105]. Recent works show that diffusion models can capture effectively complex conditional probability distributions and apply them to solve problems such as in-painting, super resolution or image-to-image translation [14, 167].

Motivated by the success of diffusion models in learning conditional distributions, recent works [155, 216] have explored applications of **conditional** diffusion models as a decoders in a VAE framework by training them to learn $p(\mathbf{x}|\mathbf{z})$. We improve upon this line of research by showing that the diffusion decoder is obsolete. Instead one can combine the encoder with an **unconditional** diffusion model via the Bayes' rule for score functions to obtain a model for $p(\mathbf{x}|\mathbf{z})$. This approach has several important advantages

1. It avoids making unrealistic Gaussian assumption on $p(\mathbf{x}|\mathbf{z})$. Therefore significantly improves the performance compared to original VAE avoiding blurry samples.

2. Since the diffusion model used in our approach is **unconditional**, i.e. it does not depend on the latent factor, our method can leverage existing powerful pre-trained diffusion models and combine them with any encoder network. Moreover the diffusion component can be always easily replaced for a better model without the need to retrain the encoder. This is in contrast to prior approaches which used **conditional** diffusion models, which have to be trained specifically to a given encoder.

3. By using the Bayes' rule for score functions we can separate training the prior from training the encoder and improve the training dynamics. This allows our method to achieve performance superior to approaches based on conditional diffusion models.

Moreover we derive a novel lower-bound on the data likelihood $p(\mathbf{x})$, which can be used to optimise the encoder in this framework.

We showcase the performance of our method on CIFAR10 [107] and CelebA [124] datasets.

6.2 Background

6.2.1 Variational Autoencoders

Variational Autoencoders (VAEs) [104, 162] is a deep latent variable model which approximates the data generating distribution $p(\mathbf{x})$. In VAEs it is assumed that the data generating process can be represented by first generating a latent variable z according to $p(\mathbf{z})$ and then generating the corresponding data point x according to the conditional data distribution $p(\mathbf{x}|\mathbf{z})$. The model is parameterized by two neural networks: the encoder e_ϕ and the decoder d_θ. VAEs make the following parametric assumptions:

- The latent prior distribution $p(\mathbf{z})$ is assumed to be a standard Gaussian $\mathcal{N}(0,I)$.
- The data posterior distribution $p_\theta(\mathbf{x}|\mathbf{z})$ is parameterised as a Gaussian $\mathcal{N}(\mu^x_\theta(\mathbf{x}),\Sigma^x_\theta(\mathbf{x}))$ where the mean $\mu^x_\theta(\mathbf{x})$ and the diagonal covariance matrix $\Sigma^x_\theta(\mathbf{x})$ are outputs of the decoder network $d_\theta : \mathbf{z} \mapsto (\mu^x_\theta(\mathbf{x}),\Sigma^x_\theta(\mathbf{x}))$. However in many implementations an additional simplification is made: $\Sigma^x_\theta(\mathbf{x})$ is assumed to be isotropic $\sigma^x_\theta I$ or even the identity matrix I.

This induces latent posterior $p_\theta(\mathbf{z}|\mathbf{x})$ and marginal likelihood distributions $p_\theta(\mathbf{x})$ which are both intractable. Therefore VAEs introduce a variational approximation

- The latent posterior distribution $p_\theta(\mathbf{z}|\mathbf{x})$ is approximated with a variational distribution $q_\phi(\mathbf{z}|\mathbf{x})$ which is parameterised as a Gaussian $\mathcal{N}(\mu^z_\phi(\mathbf{x}),\Sigma^z_\phi(\mathbf{x}))$ where the mean $\mu^z_\phi(\mathbf{x})$ and the diagonal covariance matrix $\Sigma^z_\phi(\mathbf{x})$ are outputs of the encoder network $e_\phi : \mathbf{x} \mapsto (\mu^z_\phi(\mathbf{x}),\Sigma^z_\phi(\mathbf{x}))$.

As mentioned before the exact log-likelihood $\ln p_\theta(\mathbf{x})$ is intractable, however using the variational distribution $q_\phi(\mathbf{z}|\mathbf{x})$ one obtains a tractable lower bound known as the evidence lower bound (ELBO) or variational lower bound:

$$\ln p_\theta(\mathbf{x}) - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \parallel p_\theta(\mathbf{z}|\mathbf{x})) = \mathbb{E}_{z\sim q_\phi(\mathbf{z}|\mathbf{x})}[\ln p_\theta(\mathbf{x}|\mathbf{z})] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z}))$$

Finally the model can be trained by maximizing the ELBO over all data points or by minimizing:

$$\mathscr{L}_{\text{ELBO}}(\theta,\phi) := -\mathbb{E}_{x\sim p(\mathbf{x})}\left[\mathbb{E}_{z\sim q_\phi(\mathbf{z}|\mathbf{x})}[\ln p_\theta(\mathbf{x}|\mathbf{z})] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z}))\right] \tag{6.1}$$

via a SGD based optimization method. This has the effect of both maximizing the data log-likelihood $\ln p_\theta(\mathbf{x})$ and minimizing $D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \parallel p_\theta(\mathbf{z}|\mathbf{x}))$, and therefore pushing the variational distribution $q_\phi(\mathbf{z}|\mathbf{x})$ towards the posterior $p_\theta(\mathbf{z}|\mathbf{x})$.

It is a well known problem that the KL penalty term in 6.2 leads to significant over-regularization and poor convergence. Therefore many implementations use the following modified β-ELBO objective instead [74]:

$$\mathscr{L}_{\text{ELBO}}(\theta,\phi,\beta) := -\mathbb{E}_{x\sim p(\mathbf{x})}\left[\mathbb{E}_{z\sim q_\phi(\mathbf{z}|\mathbf{x})}[\ln p_\theta(\mathbf{x}|\mathbf{z})] - \beta D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z}))\right] \tag{6.2}$$

where $\beta \in (0,1)$. The resulting model is often referred to as β-VAE [1].

6.2.2 Score-based diffusion models

Setup: In [183] score-based [85] and diffusion-based [178, 78] generative models have been unified into a single continuous-time score-based framework where the diffusion is driven by a stochastic differential equation. This framework relies on Anderson's Theorem [2], which states that under certain Lipschitz conditions on the drift coefficient $f : \mathbb{R}^{n_x} \times \mathbb{R} \to \mathbb{R}^{n_x}$ and on the diffusion coefficient $G : \mathbb{R}^{n_x} \times \mathbb{R} \to \mathbb{R}^{n_x} \times \mathbb{R}^{n_x}$ and an integrability condition on the target distribution $p(\mathbf{x}_0)$ a forward diffusion process governed by the following SDE:

$$d\mathbf{x}_t = f(\mathbf{x}_t,t)dt + G(\mathbf{x}_t,t)d\mathbf{w}_t \tag{6.3}$$

has a reverse diffusion process governed by the following SDE:

$$d\mathbf{x}_t = [f(\mathbf{x}_t,t) - G(\mathbf{x}_t,t)G(\mathbf{x}_t,t)^T\nabla_{\mathbf{x}_t}\ln p_{\mathbf{X}_t}(\mathbf{x}_t)]dt + G(\mathbf{x}_t,t)d\bar{\mathbf{w}}_t, \tag{6.4}$$

where $\bar{\mathbf{w}}_t$ is a standard Wiener process in reverse time.

The forward diffusion process transforms the *target distribution* $p(\mathbf{x}_0)$ to a *diffused distribution* $p(\mathbf{x}_T)$ after diffusion time T. By appropriately selecting the drift and the diffusion coefficients of the forward SDE, we can make sure that after sufficiently long time T, the diffused distribution $p(\mathbf{x}_T)$ approximates a simple distribution, such as $\mathscr{N}(\mathbf{0},\mathbf{I})$. We refer to this simple distribution as the *prior distribution*, denoted by π. The reverse diffusion process transforms the diffused distribution $p(\mathbf{x}_T)$ to the data distribution $p(\mathbf{x}_0)$ and the prior distribution π to a distribution p^{SDE}. The distribution p^{SDE} is close to $p(\mathbf{x}_0)$ if the diffused distribution $p(\mathbf{x}_T)$ is close to the prior distribution π. We get samples from p^{SDE} by sampling from π and simulating the reverse SDE from time T to time 0.

Sampling: To get samples by simulating the reverse SDE, we need access to the time-dependent *score function* $\nabla_{\mathbf{x}_t}\ln p(\mathbf{x}_t)$. In practice, we approximate the time-dependent score

[1] It is worth noting that choosing $\beta \neq 1$ is equivalent to choosing a fixed isotropic covariance $\Sigma^x_\theta(\mathbf{x}) = \sigma I$ for appropriate value of σ, instead of the identity matrix. We refer to [166] for details

function with a neural network $s_\theta(\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}_t} \ln p(\mathbf{x}_t)$ and simulate the reverse SDE presented in equation 6.5 to map the prior distribution π to p_θ^{SDE}.

$$d\mathbf{x}_t = [f(\mathbf{x}_t, t) - G(\mathbf{x}_t, t)G(\mathbf{x}_t, t)^T s_\theta(\mathbf{x}_t, t)]dt + G(\mathbf{x}_t, t)d\bar{\mathbf{w}}_t, \tag{6.5}$$

If the prior distribution is close to the diffused distribution and the approximated score function is close to the ground truth score function, the modeled distribution p_θ^{SDE} is provably close to the target distribution $p(\mathbf{x}_0)$. This statement is formalised in the language of distributional distances in the work of [181].

Training: A neural network $s_\theta(\mathbf{x}_t, t)$ can be trained to approximate the score function $\nabla_{\mathbf{x}_t} \ln p(\mathbf{x}_t)$ by minimizing the weighted score matching objective

$$\mathscr{L}_{SM}(\theta, \lambda(\cdot)) := \frac{1}{2}\mathbb{E}_{\substack{t\sim U(0,T) \\ \mathbf{x}_t \sim p(\mathbf{x}_t)}}[\lambda(t) \|\nabla_{\mathbf{x}_t} \ln p(\mathbf{x}_t) - s_\theta(\mathbf{x}_t, t)\|_2^2] \tag{6.6}$$

where $\lambda : [0, T] \to \mathbb{R}_+$ is a positive weighting function.

However, the above quantity cannot be optimized directly since we don't have access to the ground truth score $\nabla_{\mathbf{x}_t} \ln p(\mathbf{x}_t)$. Therefore in practice, a different objective has to be used [85, 206, 183]. In [183], the weighted denoising score-matching objective is used, which is defined as

$$\mathscr{L}_{DSM}(\theta, \lambda(\cdot)) := \frac{1}{2}\mathbb{E}_{\substack{t\sim U(0,T) \\ \mathbf{x}_0 \sim p(\mathbf{x}_0) \\ \mathbf{x}_t \sim p(\mathbf{x}_t|\mathbf{x}_0)}}[\lambda(t) \|\nabla_{\mathbf{x}_t} \ln p(\mathbf{x}_t|\mathbf{x}_0) - s_\theta(\mathbf{x}_t, t)\|_2^2] \tag{6.7}$$

The difference between DSM and SM is the replacement of the ground truth score which we do not know by the score of the perturbation kernel which we know analytically for many choices of forward SDEs. The choice of the weighted DSM objective is justified because the weighted DSM objective is equal to the SM objective up to a constant that does not depend on the parameters of the model θ. The reader can refer to [206] for the proof.

Weighting function: The choice of the weighting function is also important, because it determines the quality of score-matching in different diffusion scales. In the case of *uniform diffusion* i.e. when $G(\mathbf{x}_t, t) = g(t)I$ for $g : \mathbb{R} \to \mathbb{R}$ a principled choice for the weighting function is $\lambda(t) = g(t)^2$. This weighting function is called the likelihood weighting function [181], because it ensures that we minimize an upper bound on the Kullback–Leibler divergence from the target distribution to the model distribution by minimizing the weighted

DSM objective with this weighting. The previous statement is implied by the combination of inequality 6.8 which is proven in [181] and the relationship between the DSM and SM objectives.

$$D_{KL}(p(\mathbf{x}_0) \parallel p_\theta^{SDE}(\mathbf{x}_0)) \leq \mathscr{L}_{SM}(\theta, g(\cdot)^2) \quad + D_{KL}(p(\mathbf{x}_T) \parallel \pi) \tag{6.8}$$

Other weighting functions have also yielded very good results [101] with particular choices of forward SDEs. However, we do not have theoretical guarantees that alternative weightings would yield good results with arbitrary choices of forward SDEs.

Conditional diffusion models: The continuous score-matching framework can be extended to conditional generation, as shown in [183]. Suppose we are interested in $p(\mathbf{x}|\mathbf{z})$, where $\mathbf{x}$ is a *target data* and $\mathbf{z}$ is a *condition*. Again, we use the forward diffusion process (Equation 6.3) to obtain a family of diffused distributions $p(\mathbf{x}_t|\mathbf{z})$ and apply Anderson's Theorem to derive the *conditional reverse-time SDE*

$$dx = [f(\mathbf{x}_t, t) - G(\mathbf{x}_t, t)G(\mathbf{x}_t, t)^T \nabla_{\mathbf{x}_t} \ln p(\mathbf{x}_t|\mathbf{z})]dt + G(\mathbf{x}_t, t)d\bar{\mathbf{w}}_t. \tag{6.9}$$

Now we need to learn the conditional score $\nabla_{\mathbf{x}_t} \ln p(\mathbf{x}_t|\mathbf{z})$ in order to be able to sample from $p(\mathbf{x}|\mathbf{z})$ using reverse-time diffusion.

The conditional denoising estimator (CDE) is a way of estimating $\nabla_{\mathbf{x}_t} \ln p(\mathbf{x}_t|\mathbf{z})$ using the denoising score matching approach [206, 183]. In order to approximate $\nabla_{\mathbf{x}_t} \ln p(\mathbf{x}_t|\mathbf{z})$, the conditional denoising estimator minimizes

$$\frac{1}{2}\mathbb{E}_{\substack{t\sim U(0,T) \\ x_0,z\sim p(\mathbf{x}_0,z) \\ x_t\sim p(\mathbf{x}_t|\mathbf{x}_0)}} [\lambda(t) \|\nabla_{\mathbf{x}_t} \ln p(\mathbf{x}_t|\mathbf{x}_0) - s_\theta(\mathbf{x}_t, \mathbf{z}, t)\|_2^2]. \tag{6.10}$$

In [14] it has been shown that this is equivalent to minimizing

$$\frac{1}{2}\mathbb{E}_{\substack{t\sim U(0,T) \\ x_t,z\sim p(\mathbf{x}_t,z)}} [\lambda(t) \|\nabla_{\mathbf{x}_t} \ln p(\mathbf{x}_t|\mathbf{z}) - s_\theta(\mathbf{x}_t, \mathbf{z}, t)\|_2^2]$$

and that under mild assumptions $s_\theta(\mathbf{x}_t, \mathbf{z}, t)$ is a consistent estimator of the conditional score $\nabla_{\mathbf{x}_t} \ln p(\mathbf{x}_t|\mathbf{z})$.

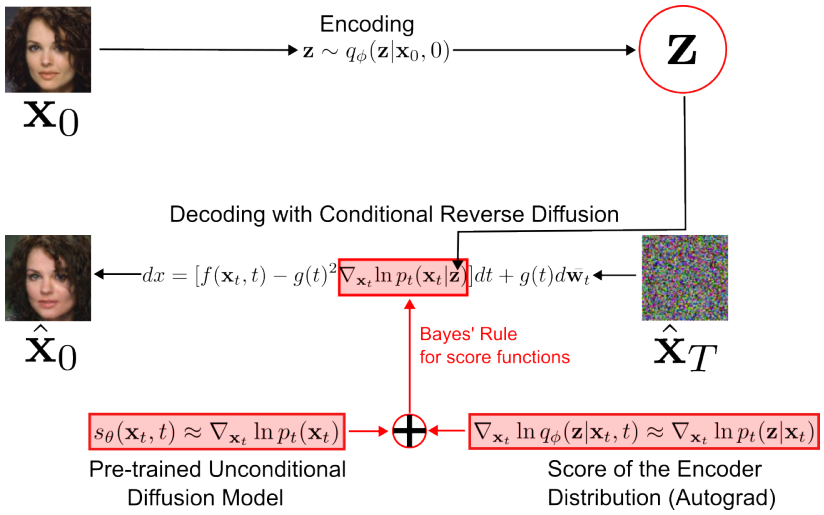


Fig. 6.1 Graphical overview of our method. The time-dependent encoder network e_ϕ induces the encoder distribution $q_\phi(\mathbf{z}|\mathbf{x}_t, t) \approx p_t(\mathbf{z}|\mathbf{x}_t)$. The data $\mathbf{x}_0$ is encoded with the encoder into a latent vector $\mathbf{z}$ by sampling $q_\phi(\mathbf{z}|\mathbf{x}_0, 0)$. Then the reconstruction $\hat{\mathbf{x}}_0$ is obtained by running the conditional reverse diffusion process using the approximate conditional data score $s_{\theta,\phi}(\mathbf{x}_t, \mathbf{z}, t) \approx \nabla_{\mathbf{x}_t} \ln p(\mathbf{x}_t|\mathbf{z})$. The model $s_{\theta,\phi}(\mathbf{x}_t, \mathbf{z}, t)$ is obtained by adding the score of unconditional diffusion model $s_\theta(\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}_t} \ln p(\mathbf{x}_t)$ and the score of the encoder distribution $\nabla_{\mathbf{x}_t} \ln q_\phi(\mathbf{z}|\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}_t} \ln p(\mathbf{z}|\mathbf{x}_t)$. The latter can be computed via automatic differentiation with respect to the input $\mathbf{x}_t$.

6.3 Method

6.3.1 Problems with conventional VAEs

As discussed before, VAEs model the conditional data distribution $p_\theta(\mathbf{x}|\mathbf{z})$ as a Gaussian distribution with mean $\mu_\theta^x(\mathbf{z})$ and covariance $\Sigma_\theta^x(\mathbf{z})$ learned by the decoder network d_θ. Moreover, in practice it is often assumed that $\Sigma_\theta^x = I$. Under these assumptions maximizing the conditional log-likelihood

$$\ln p_\theta(\mathbf{x}|\mathbf{z}) = -\frac{\left\|z - \mu_\theta^x(\mathbf{z})\right\|_2^2}{2} + \frac{d}{2} \ln 2\pi$$

is equivalent to minimizing the L_2 reconstruction error. There are several reasons why this model is inadequate when dealing with certain data modalities such as images:

- Samples from $p_\theta(\mathbf{x}|\mathbf{z})$ would look like noisy images and therefore instead of sampling the distribution $p_\theta(\mathbf{x}|\mathbf{z})$, as a principled model should, VAEs simply output the mean $\mu_\theta^x(\mathbf{z})$, which leads to undesirable smoothing and blurry samples [223].

- It is equivalent to L_2 pixel-wise error, which aligns very poorly with human perception and semantics of the image [221]. See appendix D.0.2 for a detailed discussion.

6.3.2 Conditional Diffusion Models as decoders

To mitigate above problems and avoid making the unrealistic Gaussian assumption about $p(\mathbf{x}|\mathbf{z})$ one can train a conditional diffusion model. Such approaches have been explored in [155, 216]. The conditional diffusion model $s_\theta(\mathbf{x}_t, \mathbf{z}, t)$ is trained jointly with an encoder network $e_\phi : x_0 \mapsto z$ to approximate the conditional data score $\nabla_{\mathbf{x}_t} \ln p(\mathbf{x}_t|\mathbf{z})$ by minimizing the objective 6.10. This significantly improves upon original auto-encoder framework by avoiding the Gaussian assumption and alleviates the problem of blurry samples. However, our experiments presented in section 6.4 indicate that this framework fails when trained as a Variational Autoencoder (VAE), that is, upon the introduction of the Kullback-Leibler (KL) penalty term, even when the regularization coefficient β is very small. For the remainder of this paper, we will refer to this training method as DiffDecoder.

6.3.3 Score VAE: Encoder with unconditional diffusion model as prior

In this work, we propose a further development to the above idea by leveraging Baye's rule for scores and using the structure of $\nabla_{\mathbf{x}_t} \ln p(\mathbf{x}_t|\mathbf{z})$. We can separate the training of the prior from training the encoder and improve the training dynamics. By Bayes's rule for scores we have:

$$\nabla_{\mathbf{x}_t} \ln p(\mathbf{x}_t|\mathbf{z}) = \nabla_{\mathbf{x}_t} \ln p(\mathbf{z}|\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \ln p(\mathbf{x}_t)$$

This means that we can decompose the *conditional data score* $\nabla_{\mathbf{x}_t} \ln p(\mathbf{x}_t|\mathbf{z})$ into the *data prior score* $\nabla_{\mathbf{x}_t} \ln p(\mathbf{x}_t)$ and the *latent posterior score* $\nabla_{\mathbf{x}_t} \ln p(\mathbf{z}|\mathbf{x}_t)$. The data prior score can be approximated by an unconditional diffusion model $s_\theta(\mathbf{x}_t, t)$ in the data space, allowing us to leverage powerful pretrained diffusion models. The latent posterior score $\nabla_{\mathbf{x}_t} \ln p(\mathbf{z}|\mathbf{x}_t)$ is approximated by a time-dependent encoder network $e_\phi(\mathbf{x}_t, t)$. We will discuss the details of modeling the latent posterior score in the next section. Once we have the model for data prior and latent posterior scores we can combine them to obtain the conditional data score. Then we obtain a complete latent variable model. The data $\mathbf{x}$ can be encoded to latent representation $\mathbf{z}$ using the encoder network and then it can by reconstructed by simulating the conditional reverse diffusion process using the conditional data score $\nabla_{\mathbf{x}_t} \ln p(\mathbf{x}_t|\mathbf{z})$.

This method has several advantages over the conditional diffusion approach, while preserving the benefits of having a powerful and flexible model for $p(\mathbf{x}|\mathbf{z})$. In the conditional diffusion case the score model $s_\theta(\mathbf{x}_t, \mathbf{z}, t)$ has to be trained **jointly** with the encoder network

e_ϕ. Moreover, $s_\theta(\mathbf{x}_t, \mathbf{z}, t)$ has to implicitly learn two distributions. Firstly it has to approximate $p(\mathbf{z}|\mathbf{x})$ to understand how e_ϕ encodes the information about $\mathbf{x}$ into $\mathbf{z}$ and secondly it has to model the prior $p(\mathbf{x})$ to ensure realistic reconstructions. In our approach these two tasks are clearly separated and delegated to two separate networks. Therefore, the diffusion model does not need to re-learn the encoder distribution. Instead the prior and the encoder distributions are combined in a principled analytical way via the Bayes' rule for score functions.

Moreover, the unconditional prior model $s_\theta(\mathbf{x}_t, t)$ can be trained first, **independently** of the encoder. Then we freeze the prior model and train just the encoder network. This way we always train only one network at a time, what allows for improved training dynamics.

Additionally, the data prior network can be always replaced by a better model without the need to retrain the encoder.

6.3.4 Modeling the latent posterior score $\nabla_{\mathbf{x}_t} \ln p(\mathbf{z}|\mathbf{x}_t)$

The latent posterior score is induced by the encoder network. First similarly to VAEs we impose a Gaussian parametric model at $t = 0$:

$$p_\phi(\mathbf{z}|\mathbf{x}_0) = \mathcal{N}(\mathbf{z}; \mu^z_\phi(\mathbf{x}_0), \sigma^z_\phi(\mathbf{x}_0) I)$$

where $\mu^z_\phi(\mathbf{x}_0), \sigma^z_\phi(\mathbf{x}_0)$ are the outputs of the encoder network. This together with the transition kernel $p(\mathbf{x}_t|\mathbf{x}_0)$ determines the distribution $p_\phi(\mathbf{z}|\mathbf{x}_t)$, which is given by

$$p_\phi(\mathbf{z}|\mathbf{x}_t) = \mathbb{E}_{x_0 \sim p(\mathbf{x}_0|\mathbf{x}_t)}[p_\phi(\mathbf{z}|\mathbf{x}_0)] \tag{6.11}$$

The above is computationally intractable since sampling from $p_t(\mathbf{x}_0|\mathbf{x}_t)$ would require solving the reverse SDE multiple times during each training step. Therefore we consider a variational approximation to the above distribution

$$q_{t,\phi}(\mathbf{z}|\mathbf{x}_t) = \mathcal{N}(\mathbf{z}; \mu_\phi(\mathbf{x}_t, t), \sigma_\phi(\mathbf{x}_t, t)) \tag{6.12}$$

and learn parameters ϕ such that $q_{t,\phi}(\mathbf{z}|\mathbf{x}_t) \approx p_t(\mathbf{z}|\mathbf{x}_t)$.

The choice of the above variational family is justified by the following observations:

1. At time $t = 0$ the true distribution belongs to the family, since $p(\mathbf{z}|\mathbf{x}_0)$ is Gaussian. Moreover since for small t the distribution $p_t(\mathbf{x}_0|\mathbf{x}_t)$ is very concentrated around $\mathbf{x}_0$ it is apparent from equation 6.11 that $p_\phi(\mathbf{z}|\mathbf{x}_t)$ is approximately Gaussian.

2. At time $t = 1$ the true distribution can be well approximated by a member of the variational family. This is because a noisy sample $\mathbf{x}_1$ no longer contains information

about $\mathbf{z}$, therefore $p_1(\mathbf{z}|\mathbf{x}1) \approx p(\mathbf{z})$. And since we are training with KL loss $p(\mathbf{z})$ will be approximately Gaussian.

Finally, we can use automatic differentiation to compute $\nabla_{\mathbf{x}_t} \ln q_{t,\phi}(\mathbf{z}|\mathbf{x}_t)$ which is our model for the latent posterior score $\nabla_{\mathbf{x}_t} \ln p(\mathbf{z}|\mathbf{x}_t)$.

6.3.5 Encoder Training Objective

Let $s_\theta(\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}_t} \ln p(\mathbf{x}_t)$ be a score function of a pre-trained unconditional diffusion model. Let $e_\phi : (\mathbf{x}_t, t) \mapsto (\mu^z_\phi(\mathbf{x}_t, t), \sigma^z_\phi(\mathbf{x}_t, t)I)$ be the encoder network, which defines the variational distribution $q_{t,\phi}(\mathbf{z}|\mathbf{x}_t) = \mathcal{N}\left(\mathbf{z}; \mu^z_\phi(\mathbf{x}_t, t), \sigma^z_\phi(\mathbf{x}_t, t)I\right)$ and $\nabla_{\mathbf{x}_t} \ln q_{t,\phi}(\mathbf{z}|\mathbf{x}_t)$ which approximates $\nabla_{\mathbf{x}_t} \ln p(\mathbf{z}|\mathbf{x}_t)$. By the Bayes' rule for score functions the neural approximation of the conditional data score $\nabla_{\mathbf{x}_t} \ln p(\mathbf{x}_t|\mathbf{z})$ is given by

$$s_{\theta,\phi}(\mathbf{x}_t, \mathbf{z}, t) := s_\theta(\mathbf{x}_t, t) + \nabla_{\mathbf{x}_t} \ln q_{t,\phi}(\mathbf{z}|\mathbf{x}_t)$$

We train the encoder by maximizing the marginal data log-likelihood $\ln p_{\theta,\phi}(\mathbf{x})$. In Appendix D.0.1 we show that minimizing the following training objective (with $\beta = 1$) is equivalent to maximizing the marginal data log-likelihood $\ln p_{\theta,\phi}(\mathbf{x})$,

$$\mathscr{L}_\beta(\phi) := \mathbb{E}_{\mathbf{x}_0 \sim p(\mathbf{x}_0)} \Bigg[\frac{1}{2} \mathbb{E}_{\substack{t \sim U(0,T) \\ \mathbf{x}_t \sim p_t(\mathbf{x}_t|\mathbf{x}_0) \\ \mathbf{z} \sim q_{0,\phi}(\mathbf{z}|\mathbf{x}_t)}} \left[g(t)^2 \left\| \nabla_{\mathbf{x}_t} \ln p_t(\mathbf{x}_t|\mathbf{x}_0) - s_{\theta,\phi}(\mathbf{x}_t, \mathbf{z}, t) \right\|_2^2 \right] + \beta D_{KL}\big(q_{0,\phi}(\mathbf{z}|\mathbf{x}_0) \parallel p(\mathbf{z})\big) \Bigg].$$

For the remainder of this paper, we will refer to this training method as ScoreVAE.

6.3.6 Correction of the variational error

Once the encoder e_ϕ is trained, our approximation of the ground truth decoding score $\nabla_{\mathbf{x}_t} \ln p(\mathbf{x}_t|\mathbf{z})$ is $s_{\theta,\phi}(\mathbf{x}_t, \mathbf{z}, t) := s_\theta(\mathbf{x}_t, t) + \nabla_{\mathbf{x}_t} \ln q_{t,\phi}(\mathbf{z}|\mathbf{x}_t)$. Even in the case of perfect optimisation, our approximation will not match the ground truth decoding score because of the variational approximation described in 6.12. We can correct the variational approximation error after training the encoder, by training an auxiliary correction model that approximates the residual. More specifically, we define our approximation of the decoding score to be the following: $s_{\theta,\phi}(\mathbf{x}_t, \mathbf{z}, t) := s_\theta(\mathbf{x}_t, t) + \nabla_{\mathbf{x}_t} \ln q_{t,\phi}(\mathbf{z}|\mathbf{x}_t) + c_\psi(\mathbf{x}_t, \mathbf{z}, t)$ and train the corrector model c_ψ using the same objective as in section **??** after freezing the weights of the encoder

and of the prior model which have already been trained. For the remainder of this paper, we will refer to this training method as ScoreVAE+.

6.4 Experiments

We trained our methods ScoreVAE and ScoreVAE+ on Cifar10 and CelebA 64×64 using $\beta = 0.01$. We present a quantitative comparison of our method to a β-VAE and DiffDecoder trained with the same β value in Tables 6.1 and 6.2. We present a qualitative comparison in Tables 6.3, 6.4 and a more extensive qualitative comparison in Figures D.1, D.2 in the Appendix D.2. To ensure fair comparison, we designed the models for our method, DiffDecoder and β-VAE with a very similar architecture and almost the same number of parameters. Additional experimental details are in Appendix D.1.

The DiffDecoder fails to provide consistent estimates when trained with $\beta = 0.01$ for both experiments. However, it produces consistent non-blurry reconstructions when trained as auto-encoder, i.e. with $\beta = 0$, as also shown in previous works [155, 216]. ScoreVAE outperforms both β-VAE and DiffDecoder in both experiments according to both the quantitative metrics and the qualitative results. It should be mentioned that β-VAE achieves a slightly better L_2 score in CelebA 64×64. However, β-VAE is trained to specifically minimize L_2, which is not a reliable metric for assessing the performance of VAEs (c.f. Appendix D.0.2). Both quantitative metrics and qualitative results demonstrate that correcting the variational error, as done in ScoreVAE+, provides only a slight improvement over the encoder-only method (ScoreVAE). This finding reinforces the suitability of the variational assumption.

Table 6.1 Cifar10

	L_2	LPIPS
VAE ($\beta = 0.01$)	3.410	0.269
ScoreVAE ($\beta = 0.01$)	2.634	0.125
ScoreVAE+ ($\beta = 0.01$)	**2.591**	**0.119**
DiffDecoder ($\beta = 0.01$)	19.53	0.562
DiffDecoder ($\beta = 0$)	2.851	0.127

Table 6.2 CelebA 64×64

	L_2	LPIPS
VAE ($\beta = 0.01$)	**6.97**	0.217
ScoreVAE ($\beta = 0.01$)	7.322	0.158
ScoreVAE+ ($\beta = 0.01$)	7.248	**0.155**
DiffDecoder ($\beta = 0.01$)	40.25	0.476
DiffDecoder ($\beta = 0$)	8.626	0.166

6.5 Conclusions

In this paper, we introduce a technique that enhances the Variational Auto-Encoder (VAE) framework by employing a diffusion model, thus bypassing the unrealistic Gaussian as-

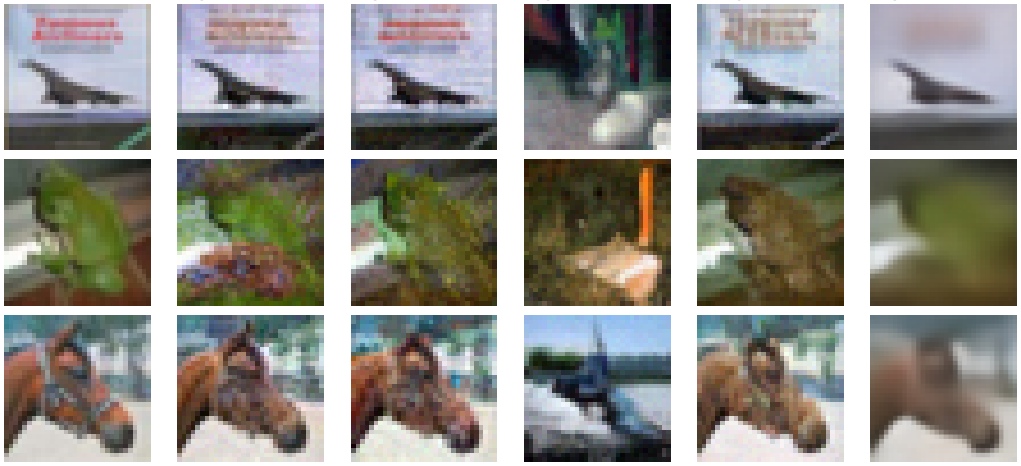


Table 6.3 Cifar10

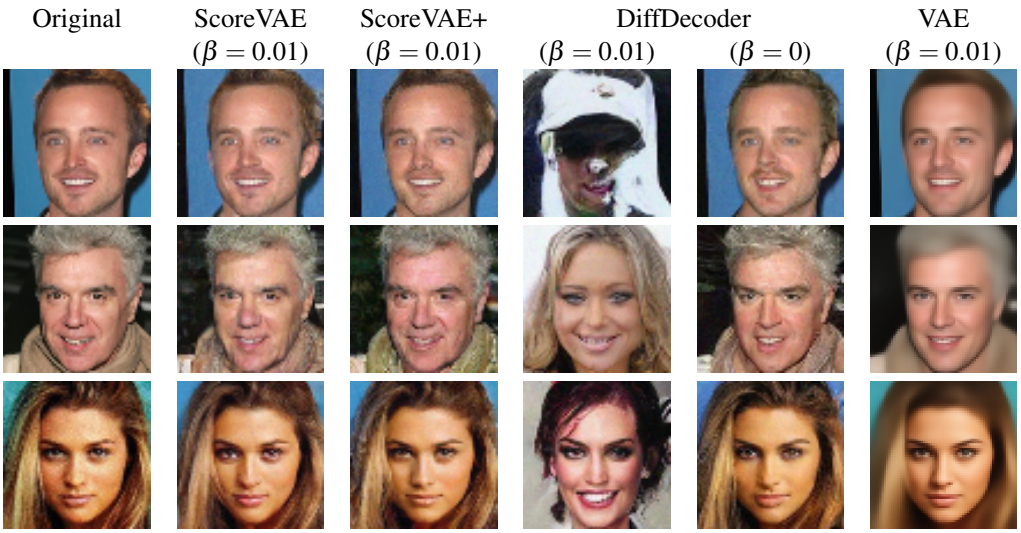


Table 6.4 CelebA 64×64

sumption inherent in the conditional data distribution $p(\mathbf{x}|\mathbf{z})$. We demonstrate using the Bayes' rule for score functions that an encoder, when paired with a pre-trained unconditional diffusion model, can result in a highly effective model for $p(\mathbf{x}|\mathbf{z})$. Thus, we show that provided that one has access to a pre-trained diffusion model for the data distribution, one can train a VAE, by training only an encoder by optimizing a novel lower-bound on the data likelihood, which we derived. Our technique outperforms the traditional β-VAE, producing clear and consistent reconstructions, free of the blurriness typically associated with the latter.

Chapter 7

Score-based pullback Riemannian geometry

Data-driven Riemannian geometry has emerged as a powerful tool for interpretable representation learning, offering improved efficiency in downstream tasks. Moving forward, it is crucial to balance cheap manifold mappings with efficient training algorithms. In this work, we integrate concepts from pullback Riemannian geometry and generative models to propose a framework for data-driven Riemannian geometry that is scalable in both geometry and learning: score-based pullback Riemannian geometry. Focusing on unimodal distributions as a first step, we propose a score-based Riemannian structure with closed-form geodesics that pass through the data probability density. With this structure, we construct a Riemannian autoencoder (RAE) with error bounds for discovering the correct data manifold dimension. This framework can naturally be used with anisotropic normalizing flows by adopting isometry regularization during training. Through numerical experiments on various datasets, we demonstrate that our framework not only produces high-quality geodesics through the data support, but also reliably estimates the intrinsic dimension of the data manifold and provides a global chart of the manifold, even in high-dimensional ambient spaces.

7.1 Introduction

Data often reside near low-dimensional non-linear manifolds as illustrated in 7.1. This manifold assumption [54] has been popular since the early work on non-linear dimension reduction [17, 36, 165, 172, 198]. Learning this non-linear structure, or representation learning, from data has proven to be highly successful [38, 102] and continues to be a

(a) Hemisphere (2,3) (b) Sinusoid (1,3) (c) Sinusoid (2,3)

Fig. 7.1 Approximate data manifolds learned by the Riemannian autoencoder generated by score-based pullback Riemannian geometry for three datasets. The orange surfaces represent the manifolds learned by the model, while the blue points correspond to the training data. Each manifold provides a convincing low-dimensional representation of the data, isometric to its respective latent space.

recurring theme in modern machine learning approaches and downstream applications [35, 59, 199, 202, 224].

Recent advances in data-driven Riemannian geometry have demonstrated its suitability for learning representations. In this context, these representations are elements residing in a learned geodesic subspace of the data space, governed by a non-trivial Riemannian structure[1] across the entire ambient space [7, 43, 70, 149, 173, 187, 194]. Among these contributions, it is worth highlighting that [187] are the first and only ones so far to use information from the full data distribution obtained though generative models [47, 183], even though this seems a natural approach given recent studies such as [170, 188]. A possible explanation for the limited use of generative models in constructing Riemannian geometry could lie in challenges regarding *scalability of the manifold mappings*. Indeed, even though the generative models can be trained efficiently, [187] also mention themselves that it can be numerically challenging to work with their induced Riemannian geometry.

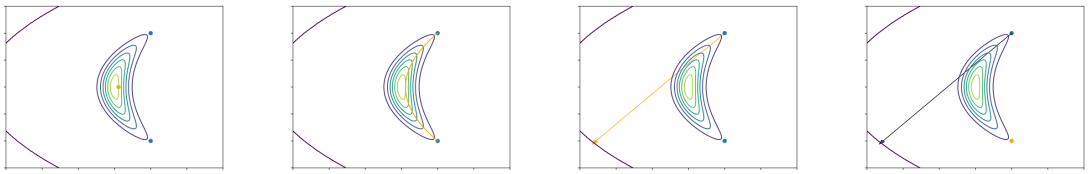
If the manifold mapping scalability challenges were to be overcome, the combined power of Riemannian geometry and state of the art generative modelling could have profound implications on how to handle data in general. Indeed, beyond typical data analysis tasks such as computing distances, means, and interpolations/extrapolations of data points as illustrated in 7.2, a data-driven Riemannian structure also offers greater potential for representation learning and downstream applications. For instance, many advanced data processing methods,

[1] rather than the standard ℓ^2-inner product

from Principal Component Analysis (PCA) to score and flow-matching, have Riemannian counterparts ([44, 55] and [31, 82]) that have proven beneficial by improving upon full black box methods in terms of interpretability [43] or Euclidean counterparts in terms of efficiency [93].

Here it is worth highlighting that scalability of manifold mappings was completely circumvented by [43] by using pullback geometry. However, here learning a suitable (and stable) pullback geometry suffers from challenges regarding *scalability of the training algorithm*, contrary to the approach by [187].

Motivated by the above, this work aims to address the following question: How to strike a good balance between scalability of training a data-driven Riemannian structure and of evaluating its corresponding manifold mappings?



(a) Barycentre (b) Geodesic (c) Logarithm (d) Exponential

Fig. 7.2 Score-based pullback Riemannian geometry (proposed) from a toy probability density.

7.1.1 Contributions

In this paper, we take first steps towards striking such a balance and propose a score-based pullback Riemannian metric assuming a relatively simple but generally applicable family of probability densities, which we show to result in both scalable manifold mappings and scalable learning algorithms. We emphasize that we do not directly aim to find the perfect balance between the two types of scalability. Instead we start from a setting which has many nice properties, but will allow for generalization to multimodal densities, which we reserve for future work.

Specifically, we consider a family of unimodal probability densities whose negative log-likelihoods are compositions of strongly convex functions and diffeomorphisms. As this work is an attempt to bridge between the geometric data analysis community and the generative modeling community, we break down the contributions in two ways. Theoretically,

- We propose a score-based pullback Riemannian metric such that manifold mappings respect the data distribution as illustrated in 7.2a,7.2b,7.2c,7.2d.

- We demonstrate that this density-based Riemannian structure naturally leads to a Riemannian autoencoder[2] and provide error bounds on the expected reconstruction error, which allows for approximation of the data manifold as illustrated in 7.1.
- We introduce a learning scheme based on adaptations of normalizing flows to find the density to be integrated into the Riemannian framework, which is tested on several synthetic data sets.

Practically, this work showcases how two simple adaptations to the normalizing flows framework enable data-driven Riemannian geometry. This significantly expands the potential for downstream applications compared to the unadapted framework.

7.1.2 Outline

After introducing notation in 7.2, 7.3 considers a family of probability distributions, from which we obtain suitable geometry, and 7.4 showcases how one can subsequently construct Riemannian Autoencoders with theoretical guarantees. From these observations 7.5 discusses the natural limitations of standard normalizing flows and how to change the parametrisation and training for downstream application in a Riemannian geometric setting. 7.6 showcases several use cases of data-driven Riemannian structure on several data sets. Finally, we summarize our findings in 7.7.

7.2 Notation

Here we present some basic notations from differential and Riemannian geometry, see [21, 27, 116, 169] for details.

Let $\mathscr{M}$ be a smooth manifold. We write $C^\infty(\mathscr{M})$ for the space of smooth functions over $\mathscr{M}$. The *tangent space* at $\mathbf{p} \in \mathscr{M}$, which is defined as the space of all *derivations* at $\mathbf{p}$, is denoted by $\mathscr{T}_\mathbf{p}\mathscr{M}$ and for *tangent vectors* we write $\Xi_\mathbf{p} \in \mathscr{T}_\mathbf{p}\mathscr{M}$. For the *tangent bundle* we write $\mathscr{T}\mathscr{M}$ and smooth vector fields, which are defined as *smooth sections* of the tangent bundle, are written as $\mathscr{X}(\mathscr{M}) \subset \mathscr{T}\mathscr{M}$.

A smooth manifold $\mathscr{M}$ becomes a *Riemannian manifold* if it is equipped with a smoothly varying *metric tensor field* $(\cdot,\cdot)\colon \mathscr{X}(\mathscr{M}) \times \mathscr{X}(\mathscr{M}) \to C^\infty(\mathscr{M})$. This tensor field induces a *(Riemannian) metric* $d_\mathscr{M}\colon \mathscr{M} \times \mathscr{M} \to \mathbb{R}$. The metric tensor can also be used to construct a unique affine connection, the *Levi-Civita connection*, that is denoted by $\nabla_{(\cdot)}(\cdot) : \mathscr{X}(\mathscr{M}) \times \mathscr{X}(\mathscr{M}) \to \mathscr{X}(\mathscr{M})$. This connection is in turn the cornerstone of a myriad of manifold

[2] in the sense of [43]

mappings. One is the notion of a *geodesic*, which for two points $\mathbf{p}, \mathbf{q} \in \mathscr{M}$ is defined as a curve $\gamma_{\mathbf{p},\mathbf{q}} : [0,1] \to \mathscr{M}$ with minimal length that connects $\mathbf{p}$ with $\mathbf{q}$. Another closely related notion to geodesics is the curve $t \mapsto \gamma_{\mathbf{p},\Xi_\mathbf{p}}(t)$ for a geodesic starting from $\mathbf{p} \in \mathscr{M}$ with velocity $\dot{\gamma}_{\mathbf{p},\Xi_\mathbf{p}}(0) = \Xi_\mathbf{p} \in \mathscr{T}_\mathbf{p}\mathscr{M}$. This can be used to define the *exponential map* $\exp_\mathbf{p} : \mathscr{D}_\mathbf{p} \to \mathscr{M}$ as

$$\exp_\mathbf{p}(\Xi_\mathbf{p}) := \gamma_{\mathbf{p},\Xi_\mathbf{p}}(1) \quad \text{where } \mathscr{D}_\mathbf{p} \subset \mathscr{T}_\mathbf{p}\mathscr{M} \text{ is the set on which } \gamma_{\mathbf{p},\Xi_\mathbf{p}}(1) \text{ is defined.} \tag{7.1}$$

Furthermore, the *logarithmic map* $\log_\mathbf{p} : \exp(\mathscr{D}'_\mathbf{p}) \to \mathscr{D}'_\mathbf{p}$ is defined as the inverse of $\exp_\mathbf{p}$, so it is well-defined on $\mathscr{D}'_\mathbf{p} \subset \mathscr{D}_\mathbf{p}$ where $\exp_\mathbf{p}$ is a diffeomorphism.

Finally, if $(\mathscr{M}, (\cdot,\cdot))$ is a d-dimensional Riemannian manifold, $\mathscr{N}$ is a d-dimensional smooth manifold and $\phi : \mathscr{N} \to \mathscr{M}$ is a diffeomorphism, the *pullback metric*

$$(\Xi, \Phi)^\phi := (D_{(\cdot)}\phi[\Xi_{(\cdot)}], D_{(\cdot)}\phi[\Phi_{(\cdot)}])_{\phi(\cdot)}, \tag{7.2}$$

where $D_\mathbf{p}\phi : \mathscr{T}_\mathbf{p}\mathscr{N} \to \mathscr{T}_{\phi(\mathbf{p})}\mathscr{M}$ denotes the differential of ϕ,

defines a Riemannian structure on $\mathscr{N}$, which we denote by $(\mathscr{N}, (\cdot,\cdot)^\phi)$. Pullback metrics literally pull back all geometric information from the Riemannian structure on $\mathscr{M}$. In particular, closed-form manifold mappings on $(\mathscr{M}, (\cdot,\cdot))$ yield under mild assumptions closed-form manifold mappings on $(\mathscr{N}, (\cdot,\cdot)^\phi)$. Throughout the rest of the paper pullback mappings will be denoted similarly to 7.2 with the diffeomorphism ϕ as a superscript, i.e., we write $d^\phi_\mathscr{N}(\mathbf{p},\mathbf{q})$, $\gamma^\phi_{\mathbf{p},\mathbf{q}}$, $\exp^\phi_\mathbf{p}(\Xi_\mathbf{p})$ and $\log^\phi_\mathbf{p}\mathbf{q}$ for $\mathbf{p},\mathbf{q} \in \mathscr{N}$ and $\Xi_\mathbf{p} \in \mathscr{T}_\mathbf{p}\mathscr{N}$.

7.3 Riemannian geometry from unimodal probability densities

We remind the reader that the ultimate goal of data-driven Riemannian geometry on $\mathbb{R}^d$ is to construct a Riemannian structure such that geodesics always pass through the support of data probability densities. In this section we will focus on constructing Riemannian geometry that does just that from unimodal densities $p : \mathbb{R}^d \to \mathbb{R}$ of the form

$$p(\mathbf{x}) \propto e^{-\psi(\varphi(\mathbf{x}))} \tag{7.3}$$

where $\psi : \mathbb{R}^d \to \mathbb{R}$ is a smooth strongly convex function and $\varphi : \mathbb{R}^d \to \mathbb{R}^d$ is a diffeomorphism, e.g., such as the density in 7.2[3]. In particular, we will consider pullback Riemannian

[3]Here, $\psi(\mathbf{x}) := 2\mathbf{x}_1^2 + \frac{1}{8}\mathbf{x}_2^2$ and $\varphi(\mathbf{x}) := (\mathbf{x}_1 - \frac{1}{9}\mathbf{x}_2^2, \mathbf{x}_2)$.

structures of the form

$$(\Xi, \Phi)_{\mathbf{x}}^{\nabla\psi\circ\varphi} := (D_{\mathbf{x}}\nabla\psi \circ \varphi[\Xi], D_{\mathbf{x}}\nabla\psi \circ \varphi[\Phi])_2, \tag{7.4}$$

which are related to the Riemannian structure obtained from the *score function* $\nabla \log(p(\cdot)) : \mathbb{R}^d \to \mathbb{R}^d$ if φ is close to a linear ℓ^2-isometry on the data support, i.e., $D_{\mathbf{x}}\varphi$ is an orthogonal operator:

$$\begin{aligned}(D_{\mathbf{x}}\nabla \log(p(\cdot))[\Xi], D_{\mathbf{x}}\nabla \log(p(\cdot))[\Phi])_2 &= (D_{\mathbf{x}}\nabla(\psi\circ\varphi)[\Xi], D_{\mathbf{x}}\nabla(\psi\circ\varphi)[\Phi])_2 \\ &= (D_{\mathbf{x}}((D_{(\cdot)}\varphi)^\top \circ \nabla\psi \circ \varphi)[\Xi], D_{\mathbf{x}}((D_{(\cdot)}\varphi)^\top \circ \nabla\psi \circ \varphi)[\Phi])_2 \\ &\approx (D_{\mathbf{x}}\nabla\psi \circ \varphi[\Xi], D_{\mathbf{x}}\nabla\psi \circ \varphi[\Phi])_2 = (\Xi, \Phi)_{\mathbf{x}}^{\nabla\psi\circ\varphi}. \end{aligned} \tag{7.5}$$

For that reason, we call such an approach to data-driven Riemannian geometry: *score-based pullback Riemannian geometry*. Since we find ourselves in a pullback setting[4], this allows to construct pullback geometry with closed-form manifold mappings.

What remains to be shown is that such geodesics and other manifold mappings pass through the data support (like in 7.2a,7.2b,7.2c,7.2d).

The following result, which is a direct application of [43, Prop. 2.1] and [43, Cor. 3.6.1], gives us closed-form expressions of several important manifold mappings under $(\cdot,\cdot)^{\nabla\psi\circ\varphi}$ and makes a connection with $(\cdot,\cdot)^{\varphi}$ if we choose

$$\psi(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{A}^{-1}\mathbf{x}, \tag{7.6}$$

where $\mathbf{A} \in \mathbb{R}^{d\times d}$ is symmetric positive definite.

This special case highlights why, in general, we obtain geodesics and manifold mappings that pass through the data support. For instance, in the scenario depicted in 7.2b, where the correct form 7.3 is used, geodesics are computed by first reversing the effect of the diffeomorphism – transforming the data distribution to resemble a Gaussian, then drawing straight lines between the morphed data points, and finally applying the diffeomorphism again. This approach results in geodesics that traverse regions of higher likelihood between the endpoints, due to the strong convexity of the quadratic function, which aligns perfectly with our objectives.

Proposition 7.3.1. *Let $\varphi : \mathbb{R}^d \to \mathbb{R}^d$ be a smooth diffeomorphism and let $\psi : \mathbb{R}^d \to \mathbb{R}$ be a smooth strongly convex function, whose Fenchel conjugate is denoted by $\psi^\star : \mathbb{R}^d \to \mathbb{R}$. Next, consider the ℓ^2-pullback manifolds $(\mathbb{R}^d, (\cdot,\cdot)^{\nabla\psi\circ\varphi})$ and $(\mathbb{R}^d, (\cdot,\cdot)^{\varphi})$ defined through metric*

[4]This is generally not true when using the score itself for probability densities of the form 7.3.

tensor fields

$$(\Xi,\Phi)_{\mathbf{x}}^{\nabla\psi\circ\varphi} := (D_{\mathbf{x}}\nabla\psi\circ\varphi[\Xi], D_{\mathbf{x}}\nabla\psi\circ\varphi[\Phi])_2, \quad \textit{and} \quad (\Xi,\Phi)_{\mathbf{x}}^{\varphi} := (D_{\mathbf{x}}\varphi[\Xi], D_{\mathbf{x}}\varphi[\Phi])_2. \tag{7.7}$$

Then,

(i) length-minimising geodesics $\gamma_{\mathbf{x},\mathbf{y}}^{\nabla\psi\circ\varphi} : [0,1] \to \mathbb{R}^d$ *on* $(\mathbb{R}^d, (\cdot,\cdot)^{\nabla\psi\circ\varphi})$ *are given by*

$$\gamma_{\mathbf{x},\mathbf{y}}^{\nabla\psi\circ\varphi}(t) = (\varphi^{-1}\circ\nabla\psi^*)((1-t)(\nabla\psi\circ\varphi)(\mathbf{x}) + t(\nabla\psi\circ\varphi)(\mathbf{y})). \tag{7.8}$$

In addition, if ψ *is of the form 7.6*

$$\gamma_{\mathbf{x},\mathbf{y}}^{\nabla\psi\circ\varphi}(t) = \gamma_{\mathbf{x},\mathbf{y}}^{\varphi}(t) = \varphi^{-1}((1-t)\varphi(\mathbf{x}) + t\varphi(\mathbf{y})). \tag{7.9}$$

(ii) the logarithmic map $\log_{\mathbf{x}}^{\nabla\psi\circ\varphi}(\cdot) : \mathbb{R}^d \to \mathscr{T}_{\mathbf{x}}\mathbb{R}^d$ *on* $(\mathbb{R}^d, (\cdot,\cdot)^{\nabla\psi\circ\varphi})$ *is given by*

$$\log_{\mathbf{x}}^{\nabla\psi\circ\varphi}\mathbf{y} = D_{\varphi(\mathbf{x})}\varphi^{-1}[D_{(\nabla\psi\circ\varphi)(\mathbf{x})}\nabla\psi^*[(\nabla\psi\circ\varphi)(\mathbf{y}) - (\nabla\psi\circ\varphi)(\mathbf{x})]]. \tag{7.10}$$

In addition, if ψ *is of the form 7.6*

$$\log_{\mathbf{x}}^{\nabla\psi\circ\varphi}\mathbf{y} = \log_{\mathbf{x}}^{\varphi}\mathbf{y} = D_{\varphi(\mathbf{x})}\varphi^{-1}[\varphi(\mathbf{y}) - \varphi(\mathbf{x})]. \tag{7.11}$$

(iii) the exponential map $\exp_{\mathbf{x}}^{\nabla\psi\circ\varphi}(\cdot) : \mathscr{T}_{\mathbf{x}}\mathbb{R}^d \to \mathbb{R}^d$ *on* $(\mathbb{R}^d, (\cdot,\cdot)^{\nabla\psi\circ\varphi})$ *is given by*

$$\exp_{\mathbf{x}}^{\nabla\psi\circ\varphi}(\Xi_{\mathbf{x}}) = (\varphi^{-1}\circ\nabla\psi^*)((\nabla\psi\circ\varphi)(\mathbf{x}) + D_{\varphi(\mathbf{x})}\nabla\psi[D_{\mathbf{x}}\varphi[\Xi_{\mathbf{x}}]]). \tag{7.12}$$

In addition, if ψ *is of the form 7.6*

$$\exp_{\mathbf{x}}^{\nabla\psi\circ\varphi}(\Xi_{\mathbf{x}}) = \exp_{\mathbf{x}}^{\varphi}(\Xi_{\mathbf{x}}) = \varphi^{-1}(\varphi(\mathbf{x}) + D_{\mathbf{x}}\varphi[\Xi_{\mathbf{x}}]). \tag{7.13}$$

(iv) the distance $d_{\mathbb{R}^d}^{\nabla\psi\circ\varphi} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ *on* $(\mathbb{R}^d, (\cdot,\cdot)^{\nabla\psi\circ\varphi})$ *is given by*

$$d_{\mathbb{R}^d}^{\nabla\psi\circ\varphi}(\mathbf{x},\mathbf{y}) = \|(\nabla\psi\circ\varphi)(\mathbf{x}) - (\nabla\psi\circ\varphi)(\mathbf{y})\|_2. \tag{7.14}$$

In addition, if ψ *is of the form 7.6*

$$d_{\mathbb{R}^d}^{\nabla\psi\circ\varphi}(\mathbf{x},\mathbf{y}) = \|\varphi(\mathbf{x}) - \varphi(\mathbf{y})\|_{\mathbf{A}^{-2}} := \|\mathbf{A}^{-1}(\varphi(\mathbf{x}) - \varphi(\mathbf{y}))\|_2. \tag{7.15}$$

(v) the Riemannian barycentre $\mathbf{x}^* \in \mathbb{R}^d$ *of the data set* $\{\mathbf{x}^i\}_{i=1}^N$ *on* $(\mathbb{R}^d, (\cdot,\cdot)^{\nabla\psi\circ\varphi})$ *is given by*

$$\mathbf{x}^* := \operatorname*{arg\,min}_{\mathbf{x}\in\mathbb{R}^d}\Big\{\frac{1}{2N}\sum_{i=1}^N d^{\nabla\psi\circ\varphi}_{\mathbb{R}^d}(\mathbf{x},\mathbf{x}^i)^2\Big\} = (\varphi^{-1}\circ\nabla\psi^*)\Big(\frac{1}{N}\sum_{i=1}^N \nabla\psi(\varphi(\mathbf{x}^i))\Big). \quad (7.16)$$

In addition, if ψ *is of the form 7.6*

$$\mathbf{x}^* := \operatorname*{arg\,min}_{\mathbf{x}\in\mathbb{R}^d}\Big\{\frac{1}{2N}\sum_{i=1}^N d^{\varphi}_{\mathbb{R}^d}(\mathbf{x},\mathbf{x}^i)^2\Big\} = \varphi^{-1}\Big(\frac{1}{N}\sum_{i=1}^N \varphi(\mathbf{x}^i)\Big). \quad (7.17)$$

Remark 7.3.2. We note that ℓ^2-stability of geodesics and the barycentre are inherited by [43, Thms. 3.4&3.8], if we have (approximate) local ℓ^2-isometry of φ on the data distribution.

7.4 Riemannian autoencoder from unimodal probability densities

The connection between $(\cdot,\cdot)^{\nabla\psi\circ\varphi}$ and $(\cdot,\cdot)^{\varphi}$ begs the question what ψ could still be used for if it is of the form 7.6. We note that this case comes down to having a data probability density that is a deformed Gaussian distribution. In the case of a regular (non-deformed) Gaussian, one can compress the data generated by it through projecting them onto a low rank approximation of the covariance matrix such that only the directions with highest variance are taken into account. This is the basic idea behind PCA. In the following we will generalize this idea to the Riemannian setting and observe that this amounts to constructing a *Riemannian autoencoder* (RAE) [43], whose error we can bound by picking the dimension of the autoencoder in a clever way, reminiscent of the classical PCA error bound.

Concretely, we assume that we have a unimodal density of the form 7.3 with a quadratic strongly convex function $\psi(\mathbf{x}) := \frac{1}{2}\mathbf{x}^\top\mathbf{A}^{-1}\mathbf{x}$ for some diagonal matrix $\mathbf{A} := \operatorname{diag}(\mathbf{a}_1,\dots\mathbf{a}_d)$ with positive entries[5]. Next, we define an indexing $u_w \in [d] := \{1,\dots,d\}$ for $w = 1,\dots,d$ such that

$$\mathbf{a}_{u_1} \geq \dots \geq \mathbf{a}_{u_d}, \quad (7.18)$$

[5]Note that this is not restrictive as for a general symmetric positive definite matrix $\mathbf{A}$ the eigenvalues can be used as diagonal entries and the orthonormal matrices can be concatenated with the diffeomorphism.

and consider a threshold $\varepsilon \in [0,1]$. We then consider $d_\varepsilon \in [d]$ defined as the integer that satisfies

$$d_\varepsilon := \begin{cases} \min\Big\{ d' \in [d-1] \;\Big|\; \sum_{w=d'+1}^{d} \mathbf{a}_{u_w} \leq \varepsilon \sum_{u=1}^{d} \mathbf{a}_u \Big\}, & \text{if } \mathbf{a}_{u_d} \leq \varepsilon \sum_{u=1}^{d} \mathbf{a}_u, \\ d, & \text{otherwise.} \end{cases}. \tag{7.19}$$

Finally, we define the mapping $E_\varepsilon : \mathbb{R}^d \to \mathbb{R}^{d_\varepsilon}$ coordinate-wise as

$$E_\varepsilon(\mathbf{x})_w := (\log^{\varphi}_{\varphi^{-1}(\mathbf{0})} \mathbf{x}, D_{\mathbf{0}}\varphi^{-1}[\mathbf{e}^{u_w}])^{\varphi}_{\varphi^{-1}(\mathbf{0})} \overset{7.11}{=} (\varphi(\mathbf{x}), \mathbf{e}^{u_w})_2, \quad w = 1, \ldots, d_\varepsilon, \tag{7.20}$$

and define $D_\varepsilon : \mathbb{R}^{d_\varepsilon} \to \mathbb{R}^d$ as

$$D_\varepsilon(\mathbf{p}) := \exp^{\varphi}_{\varphi^{-1}(\mathbf{0})}\Big(\sum_{w=1}^{d_\varepsilon} \mathbf{p}_w D_{\mathbf{0}}\varphi^{-1}[\mathbf{e}^{u_w}] \Big) \overset{7.13}{=} \varphi^{-1}\Big(\sum_{w=1}^{d_\varepsilon} \mathbf{p}_w \mathbf{e}^{u_w} \Big), \tag{7.21}$$

which generate a Riemannian autoencoder and the set $D_\varepsilon(\mathbb{R}^{d_\varepsilon}) \subset \mathbb{R}^d$ as an approximate data manifold as in the scenario in 7.1.

As hinted above, this Riemannian autoencoder comes with an error bound on the expected approximation error, which is fully determined by the diffeomorphism's deviation from isometry around the data manifold. For the proof, we refer the reader to E.1.

Theorem 7.4.1. *Let $\varphi : \mathbb{R}^d \to \mathbb{R}^d$ be a smooth diffeomorphism and let $\psi : \mathbb{R}^d \to \mathbb{R}$ be a quadratic function of the form 7.6 with diagonal $\mathbf{A} \in \mathbb{R}^{d\times d}$. Furthermore, let $p : \mathbb{R}^d \to \mathbb{R}$ be the corresponding probability density of the form 7.3. Finally, consider $\varepsilon \in [0,1]$ and the mappings $E_\varepsilon : \mathbb{R}^d \to \mathbb{R}^{d_\varepsilon}$ and $D_\varepsilon : \mathbb{R}^{d_\varepsilon} \to \mathbb{R}^d$ in 7.20,7.21 with $d_\varepsilon \in [d]$ as in 7.19.*

Then,

$$\mathbb{E}_{\mathbf{X}\sim p}[\|D_\varepsilon(E_\varepsilon(\mathbf{X})) - \mathbf{X}\|_2^2] \leq \varepsilon \inf_{\beta \in [0,\frac{1}{2})} \Big\{ \frac{C^1_{\beta,\varphi} C^2_{\beta,\varphi} C^3_{\beta,\varphi}}{1-2\beta} \Big(\frac{1+\beta}{1-2\beta} \Big)^{\frac{d}{2}} \Big\} \sum_{i=1}^{d} \mathbf{a}_i + o(\varepsilon), \tag{7.22}$$

where

$$C^1_{\beta,\varphi} := \sup_{\mathbf{x}\in\mathbb{R}^d} \{ \|D_{\varphi(\mathbf{x})}\varphi^{-1}\|_2^2 e^{-\frac{\beta}{2}\varphi(\mathbf{x})^\top \mathbf{A}^{-1}\varphi(\mathbf{x})} \}, \tag{7.23}$$

$$C^2_{\beta,\varphi} := \sup_{\mathbf{x}\in\mathbb{R}^d} \{ |\det(D_{\mathbf{x}}\varphi)| e^{-\frac{\beta}{2}\varphi(\mathbf{x})^\top \mathbf{A}^{-1}\varphi(\mathbf{x})} \}, \tag{7.24}$$

and

$$C^3_{\beta,\varphi} := \sup_{\mathbf{x}\in\mathbb{R}^d} \{ |\det(D_{\varphi(\mathbf{x})}\varphi^{-1})| e^{-\frac{\beta}{2}\varphi(\mathbf{x})^\top \mathbf{A}^{-1}\varphi(\mathbf{x})} \}. \tag{7.25}$$

Remark 7.4.2. Note that the RAE latent space is interpretable as it is ℓ^2-isometric to the data manifold if φ is an approximate ℓ^2-isometry on the data manifold. In other words, latent representations being close by or far away correspond to similar behaviour in data space, which is not the case for a VAE [102].

7.5 Learning unimodal probability densities

Naturally, we want to learn probability densities of the form 7.3, which can then directly be inserted into the proposed score-based pullback Riemannian geometry framework. In this section we will consider how to adapt normalizing flow (NF) [47] training to a setting that is more suitable for our purposes[6]. In particular, we will consider how training a normalizing flow density $p : \mathbb{R}^d \to \mathbb{R}$ given by

$$p(\mathbf{x}) := \frac{1}{C_\psi} e^{-\psi(\varphi(\mathbf{x}))} |\det(D_\mathbf{x}\varphi)|, \tag{7.26}$$

where $C_\psi > 0$ is a normalisation constant that only depends on the strongly convex function ψ, yields our target distribution 7.3.

From 7.3 and 7.4 we have seen that ideally the strongly convex function $\psi : \mathbb{R}^d \to \mathbb{R}$ corresponds to a Gaussian with a parameterised diagonal covariance matrix $\mathbf{A} \in \mathbb{R}^{d\times d}$, resulting in more parameters than in standard normalizing flows, whereas the diffeomorphism $\varphi : \mathbb{R}^d \to \mathbb{R}^d$ is regularized to be an isometry. In particular, $\mathbf{A}$ ideally allows for learnable anisotropy rather than having a fixed isotropic identity matrix. The main reason is that through anisotropy we can construct a Riemannian autoencoder (RAE), since it is known which dimensions are most important. Moreover, the diffeomorphism should be ℓ^2-isometric, unlike standard normalizing flows which are typically non-volume preserving, enabling stability (7.3.2) and a practically useful and interpretable RAE (7.4.1,7.4.2). In addition, ℓ^2-isometry (on the data support) implies volume-preservation, which means that $|\det(D_\mathbf{x}\varphi)| \approx 1$ so that 7.26 reduces to the target distribution 7.3.

This leads to learning the density through minimizing the following adapted normalizing flow loss

$$\begin{aligned}\mathscr{L}(\theta_1,\theta_2) &:= \mathbb{E}_{\mathbf{X}\sim p_{\text{data}}}\left[-\log p_{\theta_1,\theta_2}(\mathbf{X})\right] \\ &+ \lambda_{\text{vol}}\mathbb{E}_{\mathbf{X}\sim p_{\text{data}}}\left[\log(|\det\big(D_\mathbf{X}\varphi_{\theta_2}\big)|)^2\right] + \lambda_{\text{iso}}\mathbb{E}_{\mathbf{X}\sim p_{\text{data}}}\left[\|(D_\mathbf{x}\varphi_{\theta_2})^\top D_\mathbf{X}\varphi_{\theta_2} - \mathbf{I}_d\|_F^2\right]\end{aligned} \tag{7.27}$$

[6]We note that the choice for adapting the normalizing flow training scheme rather than using diffusion model training schemes is due to more robust results through the former.

where $\lambda_{\text{vol}}, \lambda_{\text{iso}} > 0$ and the negative log likelihood term reduces to

$$\begin{aligned}\mathbb{E}_{\mathbf{X}\sim p_{\text{data}}}\left[-\log p_{\theta_1,\theta_2}(\mathbf{X})\right] = \frac{1}{2}\mathbb{E}_{\mathbf{X}\sim p_{\text{data}}}\left[\varphi_{\theta_2}(\mathbf{X})^\top \mathbf{A}_{\theta_1}^{-1}\varphi_{\theta_2}(\mathbf{X})\right] \\ - \mathbb{E}_{\mathbf{X}\sim p_{\text{data}}}\left[\log\left|\det\left(D_{\mathbf{X}}\varphi_{\theta_2}\right)\right|\right] + \frac{1}{2}\operatorname{tr}(\mathbf{A}_{\theta_1}) + \frac{d}{2}\log(2\pi),\end{aligned} \tag{7.28}$$

where $\mathbf{A}_{\theta_1}$ is a diagonal matrix and φ_{θ_2} is a normalizing flow with affine coupling layers[7] [47].

7.6 Experiments

We conducted two sets of experiments to evaluate the proposed scheme from 7.5 to learn suitable pullback Riemannian geometry. The first set investigates whether our adaptation of the standard normalizing flow (NF) training paradigm leads to more accurate and stable manifold mappings, as measured by the geodesic and variation errors. The second set assesses the capability of our method to generate a robust Riemannian autoencoder (RAE).

For all experiments in this section, detailed training configurations are provided in E.4.

7.6.1 Manifold mappings

As discussed in [43], the quality of learned manifold mappings is determined by two key metrics: the *geodesic error* and the *variation error*. The geodesic error measures the average deviation form the ground truth geodesics implied by the ground truth pullback metric, while the variation error evaluates the stability of geodesics under small perturbations. We define these error metrics for the evaluation of pullback geometries in Appendix E.3.

Our approach introduces two key modifications to the normalizing flow (NF) training framework:

1. **Anisotropic Base Distribution**: We parameterize the diagonal elements of the covariance matrix $\mathbf{A}_{\theta_1}$, introducing anisotropy into the base distribution.

2. **ℓ^2-Isometry Regularization**: We regularize the flow φ_{θ_2} to be approximately ℓ^2-isometric.

[7] We note that the choice for affine coupling layers rather than using more expressive diffeomorphisms such as rational quadratic flows [50] is due to our need for high regularity for stable manifold mappings (7.3.2) and an interpretable RAE (7.4.2), which has empirically shown to be more challenging to achieve for more expressive flows as higher-order derivatives of φ will blow up the higher-order error terms in 7.4.1.

To assess the effectiveness of these modifications in learning more accurate and robust manifold mappings, we compare our method against three baselines:

(1) *Normalizing Flow (NF)*: Uses an NF with a standard isotropic Gaussian base distribution $\mathcal{N}(\mathbf{0}, \mathbf{I}_d)$ and no isometry regularization of the flow.

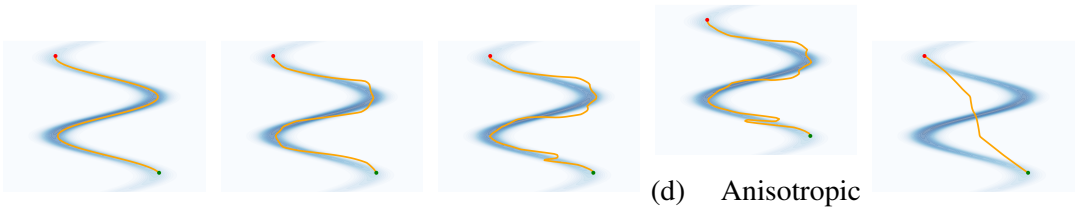
(2) *Anisotropic Normalizing Flow*: Uses an NF with the same parameterization of the diagonal covariance matrix in the base distribution as in our method, but without regularization of the flow.

(3) *Isometric Normalizing Flow*: Uses an NF with an isotropic Gaussian base distribution $\mathcal{N}(\mathbf{0}, \mathbf{I}_d)$ and regularizes the flow to be approximately ℓ^2-isometric.

We conduct experiments on three datasets, illustrated in E.1 in E.2.1: the *Single Banana Dataset*, the *Squeezed Single Banana Dataset*, and the *River Dataset*. Detailed descriptions of the construction and characteristics of these datasets are provided in E.2.1.

7.1 presents the geodesic and variation errors for each method across the three datasets and 7.3 visually compares the geodesics computed using each method on the river dataset. Our method consistently achieves significantly lower errors compared to the baselines, indicating more accurate and stable manifold mappings.

Introducing anisotropy in the base distribution without enforcing isometry in the flow offers no significant improvement over the standard flow. On the other hand, regularizing the flow to be approximately isometric without incorporating anisotropy in the base distribution results in underfitting, leading to noticeably worse performance than the standard flow. Our results demonstrate that the combination of anisotropy in the base distribution with isometry regularization (our method) yields the most accurate and stable manifold mappings, as evidenced by consistently lower geodesic and variation errors.



(a) Ground Truth (b) Our Method (c) Standard NF (d) Anisotropic NF (e) Isometric NF

Fig. 7.3 Comparison of geodesics computed using different methods on the river dataset. The geodesics generated by the proposed method have least artifacts, which is in line with our expectations from Table 7.1.

Metric	Our Method	NF	Anisotropic NF	Isometric NF
Single Banana Dataset				
Geodesic Error	**0.0315 (0.0268)**	0.0406 (0.0288)	0.0431 (0.0305)	0.0817 (0.1063)
Variation Error	**0.0625 (0.0337)**	0.0638 (0.0352)	0.0639 (0.0354)	0.0639 (0.0355)
Squeezed Single Banana Dataset				
Geodesic Error	**0.0180 (0.0226)**	0.0524 (0.0805)	0.0505 (0.0787)	0.1967 (0.2457)
Variation Error	**0.0631 (0.0326)**	0.0663 (0.0353)	0.0661 (0.0350)	0.0669 (0.0361)
River Dataset				
Geodesic Error	**0.1691 (0.0978)**	0.2369 (0.1216)	0.2561 (0.1338)	0.3859 (0.2568)
Variation Error	**0.0763 (0.0486)**	0.1064 (0.0807)	0.1113 (0.0863)	0.0636 (0.0333)

Table 7.1 Comparison of evaluation metrics for different methods across three datasets. Best-performing results for each metric are highlighted in bold. Values are reported as mean (std). The proposed method performs best in all metrics on each data set.

7.6.2 Riemannian autoencoder

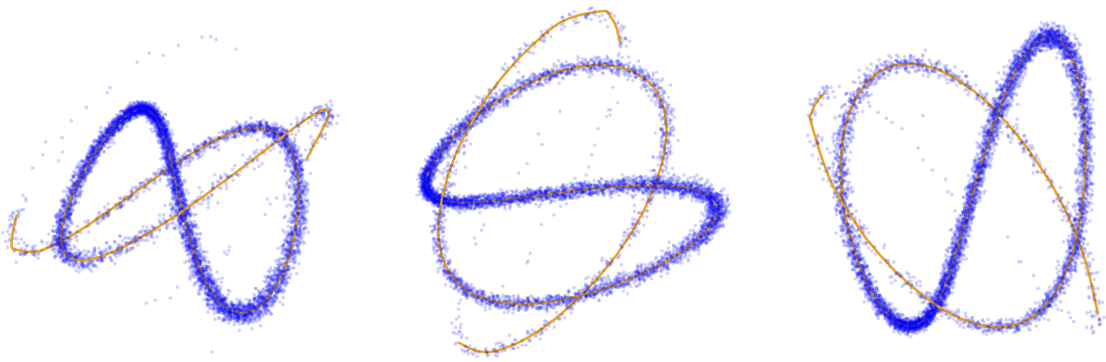
To evaluate the capacity of our method to learn a Riemannian autoencoder, we conducted experiments on two synthetic datasets across various combinations of intrinsic dimension d' and ambient dimension d:

- *Hemisphere*(d', d): Samples are drawn from the upper hemisphere of a d'-dimensional unit sphere and embedded in an d-dimensional ambient space via a random isometric mapping.
- *Sinusoid*(d', d): This dataset is generated by applying sinusoidal transformations to d'-dimensional latent variables, resulting in a complex, nonlinear manifold embedded in d dimensions.

For a detailed description of these datasets, refer to E.2.2.

1D and 2D manifolds

In 7.1 and 7.4, we present the data manifold approximations by our Riemannian autoencoder for four low-dimensional manifolds.: Hemisphere(2,3), Sinusoid(1,3), Sinusoid(2,3) and Sinusoid(1,100). In E.5, we detail the process used to create the data manifold approximations for these experiments. In our experiments, we set $\varepsilon = 0.01$, which resulted in $d_\varepsilon = d'$ for all cases, accurately capturing the intrinsic dimension of each manifold and producing accurate global charts.



(a) Dims (5, 59, 92) (b) Dims (31, 55, 66) (c) Dims (64, 72, 90)

Fig. 7.4 Approximate data manifold learned by the Riemannian autoencoder for the Sinusoid(1, 100) dataset. The orange curves depict the manifold learned by the model, while the blue points show the training data. We visualize three different combinations of the ambient dimensions.

Higher-dimensional manifolds

To evaluate the scalability of our method to higher-dimensional manifolds, we conducted additional experiments on the Hemisphere(5,20) and Sinusoid(5,20) datasets.

Our theory suggests that the learned variances indicate the importance of each latent dimension: higher variances signal more important dimensions for reconstructing the manifold, while dimensions with vanishing variances are considered insignificant and are disregarded when constructing the Riemannian autoencoder. To test the model's ability to correctly identify important and unimportant latent dimensions, we report the average ℓ^2 reconstruction error for each dataset as a function of the number of latent dimensions used. In the reconstruction error plots (see 7.5b,7.5d), we report three variance-based orders for adding latent dimensions: decreasing variance order (blue line), increasing variance order (green line), and random order (red line).

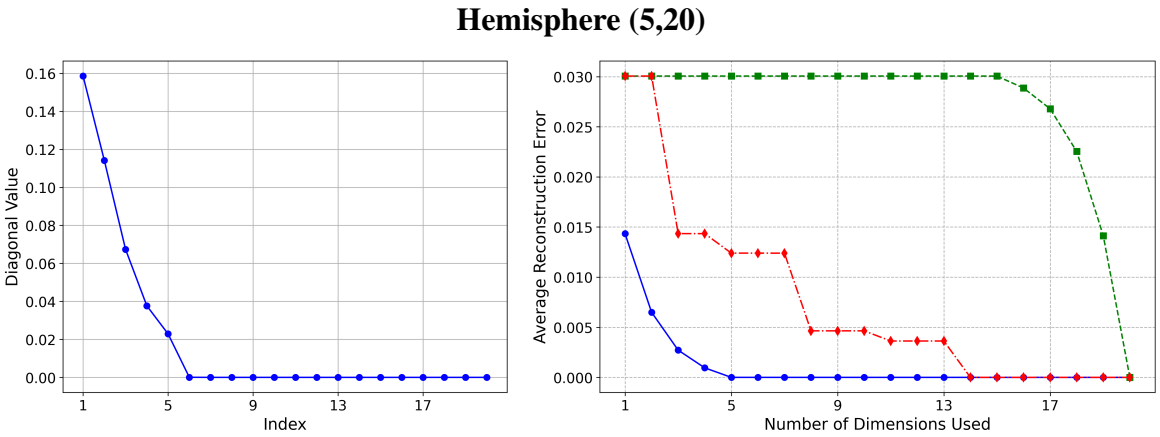
For the Hemisphere(5,20) dataset, the model identified five non-vanishing variances (see 7.5a), perfectly capturing the intrinsic dimension of the manifold. This is reflected in the blue curve in 7.5b, where the first five latent dimensions, corresponding to the largest variances, are sufficient to reduce the reconstruction error almost to zero. In contrast, the green curve illustrates that the remaining ambient dimensions do not encode useful information about the manifold. The red curve demonstrates improvement only when an important latent dimension is included.

For the more challenging Sinusoid(5,20) dataset, our method still performs very well, though not as perfectly as for the Hemisphere dataset. The first six most important latent

dimensions explain approximately 97% of the variance, increasing to over 99% with the seventh dimension (see 7.5c). This is reflected in the blue curve in 7.5d, where the first six latent dimensions reduce the reconstruction error to near zero, and the addition of the seventh dimension brings the error effectively to zero. The slight discrepancy between our results and the ground truth likely arises from increased optimization difficulty, as the normalizing flow must learn a more intricate distribution while maintaining approximate isometry. We believe that with deeper architectures and more careful tuning of the optimization loss, the model will converge to the correct intrinsic dimensionality of five. Currently, it predicts six dimensions at a threshold of $\varepsilon = 0.05$ and seven at $\varepsilon = 0.01$, slightly overestimating due to the manifold's complexity.

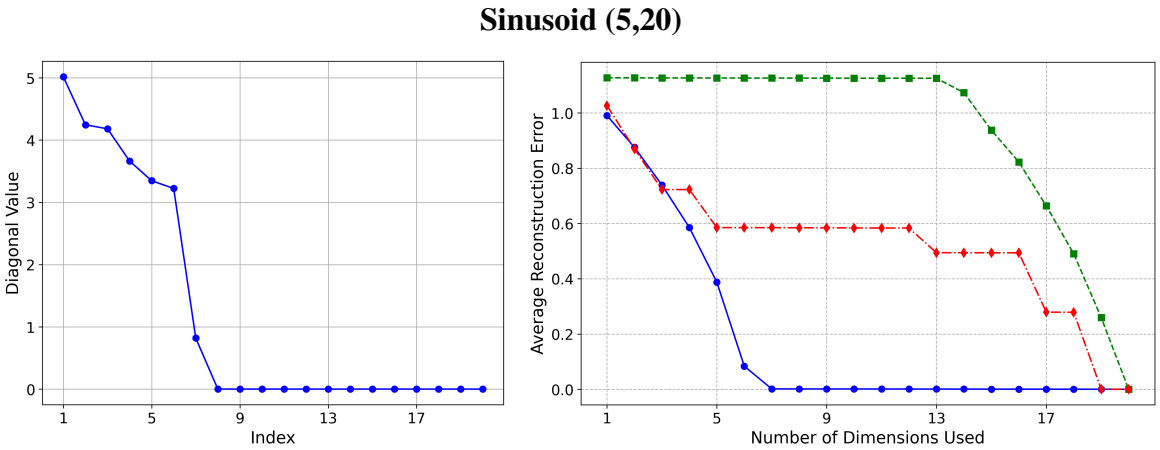
7.7 Conclusions

In this work we have taken a first step towards a practical Riemannian geometry framework for generative modelling, striking a balance between scalability of training a data-driven Riemannian structure and of evaluating its corresponding manifold mappings. We have considered a family of unimodal probability densities whose negative log-likelihoods are compositions of strongly convex functions and diffeomorphisms, and sought to learn them. We have shown that once these unimodal densities have been learned, the proposed score-based pullback geometry gives us closed-form geodesics that pass through the data probability density and a Riemannian autoencoder with error bounds that can be used to estimate the dimension of the data manifold. Finally, to learn the distribution we have proposed an adaptation to normalizing flow training. Through numerical experiments, we have shown that these modifications are crucial for extracting geometric information, and that our framework not only generates high-quality geodesics across the data support, but also accurately estimates the intrinsic dimension of the approximate data manifold while constructing a global chart, even in high-dimensional ambient spaces. Current challenges of the method lie in balancing the expressivity of the network architecture, e.g., through additional layers or more expressive architectures, and satisfying approximate ℓ^2-isometry on the data support. For future work we aim to overcome these challenges, extending the method to multimodal distributions, while making it scalable for higher-dimensional data sets.



(a) Learned variances in decreasing order.

(b) Reconstruction error for three latent orders.



(c) Learned variances in decreasing order.

(d) Reconstruction error for three latent orders.

Fig. 7.5 Learned variances and reconstruction errors for the Hemisphere(5,20) and Sinusoid(5,20) datasets. The plots in the left column show the learned variances in decreasing order for each dataset, while the right column illustrates the average ℓ^2 reconstruction error as a function of the number of latent dimensions used. The reconstruction errors are evaluated for three variance-based orders of the latent dimensions: the **blue line** (circular markers) represents adding dimensions in decreasing order of variance, the **green line** (square markers) for increasing variance, and the **red line** (diamond markers) for a random order.

Chapter 8

Conclusions and Outlook

In this thesis, we have explored several novel directions in generative modeling and generative self-supervised learning. We began with conditional multiscale autoregressive modeling, implemented using conditional normalizing flows. Building on this foundation and inspired by the efficiency and effectiveness of multiscale normalizing flows, we developed non-uniform diffusion models to replicate similar dynamics within the more powerful and rapidly emerging diffusion framework.

At that time, diffusion models began capturing our research interest, leading us to explore various intriguing ideas, such as faster sampling techniques and improving the training efficiency of Schrödinger bridges, a more general class of diffusion models. While these efforts met with partial success, they opened the door to an unexpected insight that became a turning point in our research: diffusion models inherently capture the geometry of the data manifold through the approximation of the score function.

This realization reshaped our research focus and deepened our understanding of the capabilities of generative models. We recognized that generative models like diffusion models—which do not rely on explicit assumptions about the latent structure, as VAEs or GANs do—can potentially uncover the intrinsic geometry of the data manifold. This pivotal insight fostered a profound connection between generative modeling and self-supervised learning, demonstrating that generative models can serve not only as tools for data generation but also as frameworks for understanding the underlying structure of data.

Building on this turning point, we proved that diffusion models approximate the normal bundle of the data manifold and introduced a novel method for estimating the intrinsic dimension of data manifolds. We further enhanced the modeling flexibility of vanilla β-VAEs by using a pretrained diffusion model in combination with the encoder to model the reconstruction distribution more effectively.

Finally, we introduced a new paradigm in generative modeling that goes beyond generating samples from the data distribution. This paradigm focuses on learning the underlying geometry of the data manifold, establishing a broader and more comprehensive perspective on the role of generative models.

In Chapter 3, we present *CAFLOW*, a model that improves the flexibility and expressive power of conditional flows for image-to-image translation by introducing autoregressive modeling in a scale space derived from unconditional normalizing flows. Instead of factorizing the distribution at the pixel level, CAFLOW does so across multiple scales, conditioning higher frequency details of the target image on the corresponding lower frequency content from both the target and the conditioning images. This hierarchical, frequency-based conditioning strategy leads to more effective generative modeling than standard conditional flows, as demonstrated in our experiments. Interestingly, a similar concept of autoregressive factorization in scale space has recently been explored by Tian et al. (2024) in their work on Visual Autoregressive Modeling [200]. Using powerful vision transformer architectures, their approach employs a next-scale prediction strategy and has achieved state-of-the-art performance on ImageNet at a resolution of 256×256.

In Chapter 4, we introduce *non-uniform diffusion models*, which apply a non-uniform noising schedule to different parts or scales of an image. By accelerating the diffusion of higher-frequency details while diffusing lower-frequency components more gradually, we obtain a multi-scale diffusion structure that mirrors the hierarchical design of multi-scale normalizing flows. This approach not only delivers superior image quality compared to standard uniform diffusion models, but also achieves significantly faster sampling—up to 4.4 times faster at a resolution of 128×128. We expect even greater speed-ups at higher resolutions, and believe that re-implementing this idea in modern, optimized diffusion codebases could challenge current state-of-the-art performance in image generation and sampling speed if combined with distillation methods.

Furthermore, we use non-uniform diffusion to introduce a new estimator for the conditional score—referred to here as the Conditional Multi-Speed Diffusive Estimator (CMDE), which matches the performance of the widely adopted Conditional Denoising Estimator (CDE). To our knowledge, this is the first theoretical proof of consistency for the CDE estimator, providing a solid theoretical foundation that justifies and strengthens the empirical confidence previously placed in it. Lastly, we present a detailed comparison of different methodologies for learning conditional probability distributions using score-based diffusion models, offering insights that can guide future research and practical implementations.

In Chapter 6, we introduce *ScoreVAE*, a novel approach that advances the Variational Autoencoder (VAE) framework by addressing fundamental limitations of conventional VAEs.

Traditional VAEs model the reconstruction distribution $p(x|z)$ as a Gaussian, which often leads to overly smoothed and blurry reconstructions. This limitation arises because the Gaussian assumption fails to capture the complexity and multimodality of real-world data distributions, making it difficult for the model to accurately represent intricate details and sharp features. Our method proposes a more flexible parametrization for the reconstruction distribution $p(x|z)$ that is derived from the combination of a frozen pretrained diffusion model and the learnable diffusion-time dependent encoder. Our approach also simplifies the overall training process, as it decouples the encoder from the diffusion model, allowing for flexible integration of more powerful diffusion priors without requiring full retraining which is required in diffusion decoders [155]. Future promising research directions involve scaling to higher image resolutions and explore its applicability to other data modalities such as molecular structures where powerful pretrained diffusion models are publicly available. Acquiring high-quality latent representations of such complex data could enable meaningful manipulations and more refined, controllable generative modeling, further broadening the scope and impact of ScoreVAE in the domain of representation learning.

In Chapter 5, we prove the diffusion models encode data manifolds by approximating their normal bundles. Based on this result, we propose a novel method to estimate the local intrinsic dimension (LID) of data manifolds using a trained diffusion model. This work has opened many research avenues that the community is currently exploring. Achilli et al. [1] use our LID estimation method to explore the degree of memorization in trained diffusion models. Kamkari et al. [92] use our LID estimation method to achieve state-of-the-art performance in out-of-distribution (OOD) detection. They show that likelihood based generative models can assign high likelihood to OOD samples confined to low-dimensional manifolds. They detect such cases by pairing the likelihood and the LID estimates obtained from a pretrained diffusion model, thereby robustifying OOD detection and achieving state-of-the-art performance. Kiani et al. [98] investigate the hardness of learning neural networks under the manifold hypothesis, leveraging our LID estimation method to characterize real image manifolds, and conclude that these manifolds reside within the potentially learnable regime of their proposed framework.

Inspired by the theoretical insights of Chapter 5, in Chapter 7 we propose a data-driven pullback metric based on the pseudo score function. The pseudo score function contains the same geometrical information as the score function while also being a diffeomorphism, which enables us to define a well-defined pullback Riemannian metric on the data manifold. Under certain assumptions on the form of the data distribution, this metric allows us to derive closed-form expressions for fundamental geometric constructs, such as geodesics, distances, exponential and logarithmic maps. In addition to offering closed-form expressions

for manifold maps, the constructed Riemannian structure yields a Riemannian Auto-encoder (RAE) that simultaneously learns the intrinsic dimension k of the data manifold, a global chart mapping from the data manifold $\mathscr{M}$ to $\mathbb{R}^k$ (the encoder), and the inverse chart mapping from $\mathbb{R}^k$ back to $\mathscr{M}$ (the decoder).

We implement the score-based pullback Riemannian geometry framework with Normalizing Flows. We introduce two modifications that are theoretically motivated by our analysis. We replace the fixed isotropic Gaussian base distribution with Gaussian that has a learnable covariance matrix and introduce a regularization term to keep the Normalizing Flow as close as possible to an isometry. These modifications encourage the flow to align its latent space with the true geometry of the data manifold, effectively learning an isometric embedding of the manifold into a Euclidean latent space. As a result, the learned manifold maps are more stable, more accurate, and provide a natural route to constructing a Riemannian Auto-encoder. Latent dimensions associated with high learned variance capture meaningful on-manifold degrees of freedom, while those with negligible variance represent non-informative directions and are effectively collapsed. The Normalizing Flow essentially transforms into a Riemannian Auto-encoder that captures the intrinsic dimension of the data manifold, provides the chart (as the forward map), the inverse chart (as the reverse map) and additionally provides closed form expressions for the manifold maps.

Our theory and practical implementation based on Normalizing Flows are currently guaranteed to work reliably for unimodal data distributions that are supported on or near a manifold. Interestingly, our method also performs well for multimodal data distributions. When applied to distributions that fall outside the unimodal assumption, we observe that the Riemannian Auto-encoder (RAE) tends to slightly overestimate the intrinsic dimension but still delivers good reconstructions. For instance, on the MNIST dataset, our RAE identifies 176 latent dimensions, while diffusion-based methods estimate the maximum local intrinsic dimension at approximately 152.

Looking ahead, two key avenues of exploration stand out as particularly promising for driving this work forward. The first involves refining our approach under the unimodal assumption by employing more expressive diffeomorphisms—such as rational quadratic splines [50] and free-form flows [48]—or by adapting the framework to continuous dynamical system-based methods like Flow Matching or Schrödinger bridges. These refinements have the potential to significantly improve the performance and the scalability of the framework.

The second direction focuses on developing a comprehensive theory and methodology for handling multimodal data distributions. While our current implementation, designed under the unimodal assumption, already performs well in many multimodal scenarios, achieving consistently excellent performance in such settings requires an adaptation of the framework.

Advancing this direction entails establishing a rigorous theoretical framework for geometric extraction in multimodal settings and designing an efficient, practical implementation capable of accurately capturing and leveraging the intricate geometry of complex, multimodal data distributions. These advancements will ensure the framework is robust and scalable across a broader range of data distributions.

In conclusion, this thesis forges new links between generative modeling, Riemannian geometry, and self-supervised learning. By showing how diffusion models capture the intrinsic geometric structure of data manifolds, we introduce a novel geometry-aware perspective for modeling high-dimensional data. Our data-driven pullback metric and Riemannian Autoencoder allow for principled learning of the geometry of the data manifold. Going forward, refining these methods for both unimodal and multimodal data will lead to more robust, scalable, and geometry-driven approaches to generative modeling.

References

[1] Achilli, B., Ventura, E., Silvestri, G., Pham, B., Raya, G., Krotov, D., Lucibello, C., and Ambrogioni, L. (2024). Losing dimensions: Geometric memorization in generative diffusion. *arXiv preprint arXiv:2410.08727*.

[2] Anderson, B. D. (1982). Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326.

[3] Arandjelovic, R. and Zisserman, A. (2017). Look, listen and learn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 609–617.

[4] Ardizzone, L., Kruse, J., Rother, C., and Köthe, U. (2019). Guided image generation with conditional invertible neural networks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 2395–2406.

[5] Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein GAN. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70, pages 214–223. PMLR.

[6] Arridge, S., Maass, P., Öktem, O., and Schönlieb, C.-B. (2019). Solving inverse problems using data-driven models. *Acta Numerica*, 28:1–174.

[7] Arvanitidis, G., Hansen, L. K., and Hauberg, S. (2016). A locally adaptive normal distribution. *Advances in Neural Information Processing Systems*, 29.

[8] Baek, M., DiMaio, F., Anishchenko, I., Dauparas, J., Ovchinnikov, S., Lee, G. R., Wang, J., Cong, Q., Kinch, L. N., Schaeffer, R. D., et al. (2021). Accurate prediction of protein structures and interactions using a three-track neural network. *Science*, 373(6557):871–876.

[9] Baevski, A., Zhou, Y., Mohamed, A., and Auli, M. (2020). wav2vec 2.0: A framework for self-supervised learning of speech representations. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 12449–12460.

[10] Bao, H., Dong, L., and Wei, F. (2021). Beit: Bert pre-training of image transformers. In *International Conference on Learning Representations*.

[11] Bardes, A., Ponce, J., and LeCun, Y. (2022). Vicreg: Variance-invariance-covariance regularization for self-supervised learning. In *International Conference on Learning Representations (ICLR)*.

[12] Batzolis, G., Carioni, M., Etmann, C., Afyouni, S., Kourtzi, Z., and Schönlieb, C.-B. (2024). Caflow: Conditional autoregressive flows. *Foundations of Data Science*, 6(4):553–583. The first author is supported by GSK. Early access: June 2024.

[13] Batzolis, G., Stanczuk, J., and Schönlieb, C.-B. (2023). Variational diffusion auto-encoder: Latent space extraction from pre-trained diffusion models. *arXiv preprint arXiv:2304.12141*.

[14] Batzolis, G., Stanczuk, J., Schönlieb, C.-B., and Etmann, C. (2022). Non-uniform diffusion models.

[15] Behrmann, J., Vicol, P., Wang, K.-C., Grosse, R., and Jacobsen, J.-H. (2021a). Understanding and mitigating exploding inverses in invertible neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 1792–1800. PMLR.

[16] Behrmann, J., Vicol, P., Wang, K.-C., Grosse, R., and Jacobsen, J.-H. (2021b). Understanding and mitigating exploding inverses in invertible neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 1792–1800. PMLR.

[17] Belkin, M. and Niyogi, P. (2001). Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in neural information processing systems*, 14.

[18] Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *J. Mach. Learn. Res.*, 3(null):1137–1155.

[19] Bishop, C. M. and Tipping, M. E. (2001). Probabilistic principal component analysis. PPCA.

[20] Blanch, M. G., Mrak, M., Smeaton, A. F., and O'Connor, N. E. (2019). End-to-end conditional gan-based architectures for image colourisation. In *2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–6.

[21] Boothby, W. M. (2003). *An introduction to differentiable manifolds and Riemannian geometry, Revised*, volume 120. Gulf Professional Publishing.

[22] Brehmer, J. and Cranmer, K. (2020). Flows for simultaneous manifold learning and density estimation. *Advances in Neural Information Processing Systems*, 33:442–453.

[23] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901.

[24] Butler, K. T., Davies, D. W., Cartwright, H., Isayev, O., and Walsh, A. (2018). Machine learning for molecular and materials science. *Nature*, 559(7715):547–555.

[25] Camastra, F. and Vinciarelli, A. (2002). Estimating the intrinsic dimension of data with a fractal-based method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(10):1404–1407.

[26] Campadelli, P., Casiraghi, E., Ceruti, C., and Rozza, A. (2015). Intrinsic dimension estimation: Relevant techniques and a benchmark framework. *Mathematical Problems in Engineering*, 2015:1–21.

[27] Carmo, M. P. d. (1992). *Riemannian geometry*. Birkhäuser.

[28] Caron, M., Bojanowski, P., Joulin, A., and Douze, M. (2018). Deep clustering for unsupervised learning of visual features. In *European Conference on Computer Vision (ECCV)*, pages 132–149.

[29] Caron, M., Misra, I., Mairal, J., et al. (2020). Unsupervised learning of visual features by contrasting cluster assignments. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 9912–9924.

[30] Chen, M., Huang, K., Zhao, T., and Wang, M. (2023). Score approximation, estimation and distribution recovery of diffusion models on low-dimensional data.

[31] Chen, R. T. and Lipman, Y. (2023). Riemannian flow matching on general geometries. *arXiv preprint arXiv:2302.03660*.

[32] Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. (2018). Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, volume 31.

[33] Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning (ICML)*, pages 1597–1607.

[34] Chen, X. and He, K. (2021). Exploring simple siamese representation learning. *arXiv preprint arXiv:2011.10566*.

[35] Chow, Y. L., Singh, S., Carpenter, A. E., and Way, G. P. (2022). Predicting drug polypharmacology from cell morphology readouts using variational autoencoder latent space arithmetic. *PLoS computational biology*, 18(2):e1009888.

[36] Coifman, R. R. and Lafon, S. (2006). Diffusion maps. *Applied and computational harmonic analysis*, 21(1):5–30.

[37] Cornish, R., Caterini, A., Deligiannidis, G., and Doucet, A. (2020). Relaxing bijectivity constraints with continuously indexed normalising flows. In *International conference on machine learning*, pages 2133–2143. PMLR.

[38] DeMers, D. and Cottrell, G. (1992). Non-linear dimensionality reduction. *Advances in neural information processing systems*, 5.

[39] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255.

[40] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019a). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

[41] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019b). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[42] Dhariwal, P. and Nichol, A. (2021). Diffusion models beat gans on image synthesis.

[43] Diepeveen, W. (2024). Pulling back symmetric riemannian geometry for data analysis. *arXiv preprint arXiv:2403.06612*.

[44] Diepeveen, W., Chew, J., and Needell, D. (2023). Curvature corrected tangent space-based approximation of manifold-valued data. *arXiv preprint arXiv:2306.00507*.

[45] Ding, Y., Hu, H., Ge, Z., Wu, J., Zhang, H., and Ding, C. (2021). Vq-vae for multimodal image synthesis. *arXiv preprint arXiv:2103.02398*.

[46] Dinh, L., Krueger, D., and Bengio, Y. (2015). NICE: non-linear independent components estimation. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015*.

[47] Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2017). Density estimation using real NVP. In *International Conference on Learning Representations*.

[48] Draxler, F., Sorrenson, P., Zimmermann, L., Rousselot, A., and Köthe, U. (2024). Free-form flows: Make any architecture a normalizing flow. In *International Conference on Artificial Intelligence and Statistics*, pages 2197–2205. PMLR.

[49] Du, Y. and Mordatch, I. (2019). Implicit generation and modeling with energy-based models. In *Advances in Neural Information Processing Systems*, volume 32.

[50] Durkan, C., Bekasov, A., Murray, I., and Papamakarios, G. (2019). Neural spline flows. *Advances in neural information processing systems*, 32.

[51] Esser, P., Rombach, R., and Ommer, B. (2021). Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12873–12883.

[52] Fan, M., Gu, N., Qiao, H., and Zhang, B. (2010). Intrinsic dimension estimation of data by principal component analysis.

[53] Fefferman, C., Mitter, S., and Narayanan, H. (2013). Testing the manifold hypothesis.

[54] Fefferman, C., Mitter, S., and Narayanan, H. (2016). Testing the manifold hypothesis. *Journal of the American Mathematical Society*, 29(4):983–1049.

[55] Fletcher, P. T., Lu, C., Pizer, S. M., and Joshi, S. (2004). Principal geodesic analysis for the study of nonlinear statistics of shape. *IEEE transactions on medical imaging*, 23(8):995–1005.

[56] Fukunaga, K. and Olsen, D. (1971). An algorithm for finding intrinsic dimensionality of data. *IEEE Transactions on Computers*, C-20(2):176–183.

[57] Gentile, F., Agrawal, V., Hsing, M.-F., Ton, A.-T., Ban, F., Norinder, U., Gleave, M. E., and Cherkasov, A. (2020). Deep docking: A deep learning platform for ultra-large virtual screening. *Journal of Chemical Information and Modeling*, 60(9):4291–4305.

[58] Gidaris, S., Singh, P., and Komodakis, N. (2018). Unsupervised representation learning by predicting image rotations. In *International Conference on Learning Representations (ICLR)*.

[59] Gomari, D. P., Schweickart, A., Cerchietti, L., Paietta, E., Fernandez, H., Al-Amin, H., Suhre, K., and Krumsiek, J. (2022). Variational autoencoders learn transferrable representations of metabolomics data. *Communications Biology*, 5(1):645.

[60] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT Press.

[61] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014a). Generative adversarial nets. In *Advances in Neural Information Processing Systems*, volume 27.

[62] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014b). Generative adversarial networks.

[63] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014c). Generative adversarial networks.

[64] Goyal, A. and Bengio, Y. (2022). Inductive biases for deep learning of higher-level cognition. *Proceedings of the Royal Society A*, 478(2266):20210068.

[65] Goyal, P., Mahajan, D., Gupta, A., and Misra, I. (2021). Self-supervised pretraining of visual features in the wild. *arXiv preprint arXiv:2103.01988*.

[66] Grathwohl, W., Wang, K. C., Jacobsen, J.-H., Duvenaud, D., Swersky, K., and Norouzi, M. (2020). Your classifier is secretly an energy-based model and you should treat it like one. In *International Conference on Learning Representations (ICLR)*.

[67] Grill, J.-B., Strub, F., Altché, F., et al. (2020). Bootstrap your own latent: A new approach to self-supervised learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 21271–21284.

[68] Grover, A., Chute, C., Shu, R., Cao, Z., and Ermon, S. (2020). Alignflow: Cycle consistent learning from multiple domains via normalizing flows. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4028–4035.

[69] Haro, G., Randall, G., and Sapiro, G. (2008). Translated poisson mixture model for stratification learning. *Int. J. Comput. Vis.*, 80(3):358–374.

[70] Hauberg, S., Freifeld, O., and Black, M. (2012). A geometric take on metric learning. *Advances in Neural Information Processing Systems*, 25.

[71] He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. (2022). Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009.

[72] He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9729–9738.

[73] Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. (2018). Gans trained by a two time-scale update rule converge to a local nash equilibrium.

[74] Higgins, I., Matthey, L., Pal, A., Burgess, C. P., Glorot, X., Botvinick, M. M., Mohamed, S., and Lerchner, A. (2016). beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*.

[75] Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800.

[76] Ho, J., Chen, X., Srinivas, A., Duan, Y., and Abbeel, P. (2019). Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *International Conference on Machine Learning*, pages 2722–2730. PMLR.

[77] Ho, J., Jain, A., and Abbeel, P. (2020a). Denoising diffusion probabilistic models.

[78] Ho, J., Jain, A., and Abbeel, P. (2020b). Denoising diffusion probabilistic models.

[79] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

[80] Horvat, C. and Pfister, J.-P. (2022). Intrinsic dimensionality estimation using normalizing flows. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A., editors, *Advances in Neural Information Processing Systems*, volume 35, pages 12225–12236. Curran Associates, Inc.

[81] Hossain, M., Sohel, F., Shiratuddin, M. F., and Laga, H. (2020). A comprehensive survey of image caption generation techniques. *Journal of Visual Communication and Image Representation*, 71:102844.

[82] Huang, C.-W., Aghajohari, M., Bose, J., Panangaden, P., and Courville, A. C. (2022). Riemannian diffusion models. *Advances in Neural Information Processing Systems*, 35:2750–2761.

[83] Huang, C.-W., Krueger, D., Lacoste, A., and Courville, A. (2018). Neural autoregressive flows. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 2078–2087.

[84] Hyvärinen, A. (2005a). Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(24):695–709.

[85] Hyvärinen, A. (2005b). Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(24):695–709.

[86] Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134.

[87] Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2018). Image-to-image translation with conditional adversarial networks.

[88] Izmailov, P., Kirichenko, P., Finzi, M., and Wilson, A. G. (2021). Semi-supervised learning with normalizing flows. In *International Conference on Learning Representations*.

[89] Jaini, P., Kobyzev, I., Yu, Y., and Brubaker, M. (2020). Tails of lipschitz triangular flows. In *International Conference on Machine Learning*, pages 4673–4681. PMLR.

[90] Johnsson, K. (2016). intrinsicdimension: Intrinsic dimension estimation.

[91] Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., et al. (2021). Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589.

[92] Kamkari, H., Ross, B. L., Cresswell, J. C., Caterini, A. L., Krishnan, R. G., and Loaiza-Ganem, G. (2024). A geometric explanation of the likelihood ood detection paradox. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, Vienna, Austria. PMLR. Copyright 2024 by the author(s).

[93] Kapusniak, K., Potaptchik, P., Reu, T., Zhang, L., Tong, A., Bronstein, M., Bose, A. J., and Giovanni, F. D. (2024). Metric flow matching for smooth interpolations on the data manifold.

[94] Karras, T., Laine, S., and Aila, T. (2019). A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4401–4410.

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

[96] Kégl, B. (2002). Intrinsic dimension estimation using packing numbers. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems*, volume 15. MIT Press.

[97] Keys, R. (1981). Cubic convolution interpolation for digital image processing. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 29(6):1153–1160.

[98] Kiani, B. T., Wang, J., and Weber, M. (2024). Hardness of learning neural networks under the manifold hypothesis. In *Advances in Neural Information Processing Systems (NeurIPS)*. Spotlight Presentation.

[99] Kim, J., Shin, J., Rinaldo, A., and Wasserman, L. (2019). Uniform convergence rate of the kernel density estimator adaptive to intrinsic volume dimension. In *International Conference on Machine Learning*, pages 3398–3407. PMLR.

[100] Kingma, D. P. and Dhariwal, P. (2018). Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, volume 31, pages 10215–10224.

[101] Kingma, D. P., Salimans, T., Poole, B., and Ho, J. (2021). Variational diffusion models.

[102] Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

[103] Kingma, D. P. and Welling, M. (2014a). Auto-encoding variational bayes.

[104] Kingma, D. P. and Welling, M. (2014b). Auto-encoding variational bayes. In Bengio, Y. and LeCun, Y., editors, *2nd International Conference on Learning Representations, ICLR 2014*.

[105] Kong, Z., Ping, W., Huang, J., Zhao, K., and Catanzaro, B. (2020). Diffwave: A versatile diffusion model for audio synthesis.

[106] Kpotufe, S. (2011). k-nn regression adapts to local intrinsic dimension. *Advances in neural information processing systems*, 24.

[107] Krizhevsky, A. (2012). Learning multiple layers of features from tiny images. *University of Toronto*.

[108] Langevin, P. (1908). Sur la théorie du mouvement brownien. *Comptes Rendus de l'Académie des Sciences (Paris)*, 146:530–533.

[109] Laszkiewicz, M., Lederer, J., and Fischer, A. (2021). Copula-based normalizing flows. *arXiv preprint arXiv:2107.07352*.

[110] LeCun, Y. (2021). Self-supervised learning: The dark matter of intelligence. *Facebook Research*.

[111] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

[112] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[113] LeCun, Y., Chopra, S., and Hadsell, R. (2006). A tutorial on energy-based learning. *Predicting Structured Data*.

[114] LeCun, Y. and Cortes, C. (2010). MNIST handwritten digit database.

[115] Lee, J. (2019). *Introduction to Riemannian Manifolds*. Graduate Texts in Mathematics. Springer International Publishing.

[116] Lee, J. M. (2013). Smooth manifolds. In *Introduction to Smooth Manifolds*, pages 1–31. Springer.

[117] Levina, E. and Bickel, P. (2004). Maximum likelihood estimation of intrinsic dimension. In Saul, L., Weiss, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems*, volume 17. MIT Press.

[118] Li, Z., Yang, J., Liu, Z., Yang, X., Jeon, G., and Wu, W. (2019). Feedback network for image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3867–3876.

[119] Liang, J., Lugmayr, A., Zhang, K., Danelljan, M., Gool, L. V., and Timofte, R. (2021a). Hierarchical conditional flow: A unified framework for image super-resolution and image rescaling.

[120] Liang, J., Lugmayr, A., Zhang, K., Danelljan, M., Van Gool, L., and Timofte, R. (2021b). Hierarchical conditional flow: A unified framework for image super-resolution and image rescaling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4076–4085.

[121] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

[122] Liu, Z., Luo, P., Wang, X., and Tang, X. (2015a). Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pages 3730–3738.

[123] Liu, Z., Luo, P., Wang, X., and Tang, X. (2015b). Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*.

[124] Liu, Z., Luo, P., Wang, X., and Tang, X. (2015c). Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*.

[125] Lu, Y. and Huang, B. (2020). Structured output learning with conditional generative flows. *AAAI*.

[126] Lugmayr, A., Danelljan, M., Romero, A., Sabater, N., Timofte, R., and Van Gool, L. (2022). Repaint: Inpainting using denoising diffusion probabilistic models. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11461–11471.

[127] Lugmayr, A., Danelljan, M., Van Gool, L., and Timofte, R. (2020). Srflow: Learning the super-resolution space with normalizing flow. In *Computer Vision – ECCV 2020*.

[128] Léonard, C. (2013). Some properties of path measures.

[129] Marinescu, R. V., Moyer, D., and Golland, P. (2021). Bayesian image reconstruction using deep generative models.

[130] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. In *International Conference on Learning Representations (ICLR)*.

[131] Minka, T. (2000). Automatic choice of dimensionality for pca. In Leen, T., Dietterich, T., and Tresp, V., editors, *Advances in Neural Information Processing Systems*, volume 13. MIT Press.

[132] Misra, I., Zitnick, C. L., and Hebert, M. (2016). Shuffle and learn: unsupervised learning using temporal order verification. In *European Conference on Computer Vision (ECCV)*, pages 527–544. Springer.

[133] Mueller, J. L. and Siltanen, S. (2012). Linear and nonlinear inverse problems with practical applications. In *Computational science and engineering*.

[134] Neal, R. M. (2011). MCMC Using Hamiltonian Dynamics. In Brooks, S., Gelman, A., Jones, G. L., and Meng, X.-L., editors, *Handbook of Markov Chain Monte Carlo*, pages 113–162. Chapman and Hall/CRC, Boca Raton, FL.

[135] Newey, W. K. and McFadden, D. (1994). Chapter 36 large sample estimation and hypothesis testing. volume 4 of *Handbook of Econometrics*, pages 2111–2245. Elsevier.

[136] Nicolaescu, L. (2011). *An Invitation to Morse Theory*. Universitext. Springer New York.

[137] Noroozi, M. and Favaro, P. (2016). Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision (ECCV)*, pages 69–84. Springer.

[138] Oko, K., Akiyama, S., and Suzuki, T. (2023). Diffusion models are minimax optimal distribution estimators.

[139] Oksendal, B. (2003). *Stochastic Differential Equations (5th Ed.): An Introduction with Applications*. Springer-Verlag, Heidelberg.

[140] Onken, D., Fung, S. W., Li, X., and Ruthotto, L. (2021). Ot-flow: Fast and accurate continuous normalizing flows via optimal transport. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9223–9232.

[141] Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. (2016a). Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*.

[142] Oord, A. v. d., Kalchbrenner, N., and Kavukcuoglu, K. (2016b). Pixel recurrent neural networks. In *International Conference on Machine Learning (ICML)*, pages 1747–1756.

[143] Oord, A. v. d., Li, Y., and Vinyals, O. (2018). Representation learning with contrastive predictive coding. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 10256–10265.

[144] Oord, A. v. d., Vinyals, O., and Kavukcuoglu, K. (2017). Neural discrete representation learning. *Advances in Neural Information Processing Systems (NeurIPS)*, pages 6306–6315.

[145] Palais, R. S. and Terng, C. (1988). *Critical Point Theory and Submanifold Geometry*. Critical Point Theory and Submanifold Geometry. Springer.

[146] Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. (2021a). Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64.

[147] Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. (2021b). Normalizing flows for probabilistic modeling and inference.

[148] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

[149] Peltonen, J., Klami, A., and Kaski, S. (2004). Improved learning of riemannian metrics for exploratory analysis. *Neural Networks*, 17(8-9):1087–1100.

[150] Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

[151] Pettis, K. W., Bailey, T. A., Jain, A. K., and Dubes, R. C. (1979). An intrinsic dimensionality estimator from near-neighbor information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(1):25–37.

[152] Pidstrigach, J. (2022). Score-based generative models detect manifolds.

[153] Poggio, T., Mhaskar, H., Rosasco, L., Miranda, B., and Liao, Q. (2017). Why and when can deep–but not shallow–networks avoid the curse of dimensionality: A review. *International Journal of Automation and Computing*, 14(5):503–519.

[154] Pope, P., Zhu, C., Abdelkader, A., Goldblum, M., and Goldstein, T. (2021). The intrinsic dimension of images and its impact on learning. *arXiv preprint arXiv:2104.08894*.

[155] Preechakul, K., Chatthee, N., Wizadwongsa, S., and Suwajanakorn, S. (2022). Diffusion autoencoders: Toward a meaningful and decodable representation.

[156] Pumarola, A., Popov, S., Moreno-Noguer, F., and Ferrari, V. (2020). C-flow: Conditional generative flow models for images and 3d point clouds. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7946–7955.

[157] Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. (2021). Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR.

[158] Radford, A., Wu, J., Amodei, D., et al. (2019). Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

[159] Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I. (2021). Zero-shot text-to-image generation. In *International conference on machine learning*, pages 8821–8831. Pmlr.

[160] Razavi, A., van den Oord, A., and Vinyals, O. (2019). Generating diverse high-fidelity images with vq-vae-2. In *Advances in Neural Information Processing Systems*, pages 14866–14876.

[161] Rezende, D. and Mohamed, S. (2015). Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538. PMLR.

[162] Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models.

[163] Rives, A., Meier, J., Sercu, T., Goyal, S., Lin, Z., Liu, J., Guo, D., Ott, M., Zitnick, C. L., Ma, J., and Fergus, R. (2021). Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 118(15).

[164] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695.

[165] Roweis, S. T. and Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326.

[166] Rybkin, O., Daniilidis, K., and Levine, S. (2021). Simple and effective vae training with calibrated decoders.

[167] Saharia, C., Ho, J., Chan, W., Salimans, T., Fleet, D. J., and Norouzi, M. (2021). Image super-resolution via iterative refinement.

[168] Saharia, C., Ho, J., Chan, W., Salimans, T., Fleet, D. J., and Norouzi, M. (2022). Image super-resolution via iterative refinement. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[169] Sakai, T. (1996). *Riemannian geometry*, volume 149. American Mathematical Soc.

[170] Sakamoto, K., Tanabe, M., Akagawa, M., Hayashi, Y., Sakamoto, R., Yaguchi, M., Suzuki, M., and Matsuo, Y. (2024). The geometry of diffusion models: Tubular neighbourhoods and singularities. In *ICML 2024 Workshop on Geometry-grounded Representation Learning and Generative Modeling*.

[171] Salimans, T. and Ho, J. (2022). Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*.

[172] Sammon, J. W. (1969). A nonlinear mapping for data structure analysis. *IEEE Transactions on computers*, 100(5):401–409.

[173] Scarvelis, C. and Solomon, J. (2023). Riemannian metric learning via optimal transport. In *The Eleventh International Conference on Learning Representations*.

[174] Schneider, S., Baevski, A., Collobert, R., Auli, M., and Mohamed, A. (2019). wav2vec: Unsupervised pre-training for speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6415–6419.

[175] Scott, D. W. (2015). *Multivariate Density Estimation: Theory, Practice, and Visualization*. John Wiley & Sons.

[176] Silverman, B. W. (1986). *Density Estimation for Statistics and Data Analysis*. Chapman and Hall.

[177] Sohl-Dickstein, J., Weiss, E. A., Maheswaranathan, N., and Ganguli, S. (2015a). Deep unsupervised learning using nonequilibrium thermodynamics.

[178] Sohl-Dickstein, J., Weiss, E. A., Maheswaranathan, N., and Ganguli, S. (2015b). Deep unsupervised learning using nonequilibrium thermodynamics.

[179] Song, J., Meng, C., and Ermon, S. (2020a). Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*.

[180] Song, Y., Dhariwal, P., Chen, M., and Sutskever, I. (2023). Consistency models. *arXiv preprint arXiv:2303.01469*.

[181] Song, Y., Durkan, C., Murray, I., and Ermon, S. (2021a). Maximum likelihood training of score-based diffusion models.

[182] Song, Y. and Ermon, S. (2020). Generative modeling by estimating gradients of the data distribution.

[183] Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2020b). Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*.

[184] Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2020c). Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*.

[185] Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2021b). Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*.

[186] Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2021c). Score-based generative modeling through stochastic differential equations.

[187] Sorrenson, P., Behrend-Uriarte, D., Schnörr, C., and Köthe, U. (2024). Learning distances from data with normalizing flows and score matching.

[188] Stanczuk, J., Batzolis, G., Deveney, T., and Schönlieb, C.-B. (2022). Your diffusion model secretly knows the dimension of the data manifold. *arXiv preprint arXiv:2212.12611*.

[189] Stanczuk, J., Etmann, C., Kreusser, L. M., and Schönlieb, C.-B. (2021). Wasserstein gans work because they fail (to approximate the wasserstein distance).

[190] Stanczuk, J. P., Batzolis, G., Deveney, T., and Schönlieb, C.-B. (2024). Diffusion models encode the intrinsic dimension of data manifolds. In Salakhutdinov, R., Kolter, Z., Heller, K., Weller, A., Oliver, N., Scarlett, J., and Berkenkamp, F., editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 46412–46440. PMLR.

[191] Stokes, J. M., Yang, K., Swanson, K., Jin, W., Cubillos-Ruiz, J. R., Donghia, N. M., MacNair, C. R., French, S., Carfrae, L. A., Bloom-Ackermann, Z., et al. (2020). A deep learning approach to antibiotic discovery. *Cell*, 180(4):688–702.

[192] Sun, C., Myers, A., Vondrick, C., Murphy, K., and Schmid, C. (2019a). Videobert: A joint model for video and language representation learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7464–7473.

[193] Sun, H., Mehta, R., Zhou, H. H., Huang, Z., Johnson, S. C., Prabhakaran, V., and Singh, V. (2019b). Dual-glow: Conditional flow-based generative model for modality transfer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

[194] Sun, X., Liao, D., MacDonald, K., Zhang, Y., Huguet, G., Wolf, G., Adelstein, I., Rudner, T. G. J., and Krishnaswamy, S. (2024). Geometry-aware autoencoders for metric learning and generative modeling on data manifolds. In *ICML 2024 Workshop on Geometry-grounded Representation Learning and Generative Modeling*.

[195] Suvorov, R., Logacheva, E., Mashikhin, A., Remizova, A., Ashukha, A., Korzhenkov, D., Galyautdinov, I., Kong, N., Goka, H., and Lempitsky, V. (2022). Resolution-robust large mask inpainting with fourier convolutions. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8564–8573.

[196] Tashiro, Y., Song, J., Song, Y., and Ermon, S. (2021). Csdi: Conditional score-based diffusion models for probabilistic time series imputation.

[197] Tempczyk, P., Michaluk, R., Łukasz Garncarek, Spurek, P., Tabor, J., and Goliński, A. (2022). Lidl: Local intrinsic dimension estimation using approximate likelihood.

[198] Tenenbaum, J. B., Silva, V. d., and Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323.

[199] Ternes, L., Dane, M., Gross, S., Labrie, M., Mills, G., Gray, J., Heiser, L., and Chang, Y. H. (2022). A multi-encoder variational autoencoder controls multiple transformational features in single-cell image analysis. *Communications biology*, 5(1):255.

[200] Tian, K., Jiang, Y., Yuan, Z., Peng, B., and Wang, L. (2024). Visual autoregressive modeling: Scalable image generation via next-scale prediction. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[201] Tian, Y. et al. (2022). Understanding self-supervised contrastive learning with generalized contrastive losses. *arXiv preprint arXiv:2202.09671*.

[202] Vahdat, A. and Kautz, J. (2020). Nvae: A deep hierarchical variational autoencoder. *Advances in neural information processing systems*, 33:19667–19679.

[203] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

[204] Verine, A., Negrevergne, B., Chevaleyre, Y., and Rossi, F. (2023). On the expressivity of bi-lipschitz normalizing flows. In *Asian Conference on Machine Learning*, pages 1054–1069. PMLR.

[205] Viazovetskyi, Y., Ivashkin, V., and Kashin, E. (2020). Stylegan2 distillation for feed-forward image manipulation. In *European Conference on Computer Vision*, pages 170–186. Springer.

[206] Vincent, P. (2011). A connection between score matching and denoising autoencoders. *Neural Computation*, 23(7):1661–1674.

[207] Wang, R., Wei, F., Sun, C., Murphy, K., and Schmid, C. (2023). Imagebind: Learning joint representations of vision, audio, and language without supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2256–2266.

[208] Wang, X., Yu, K., Wu, S., Gu, J., Liu, Y., Dong, C., Qiao, Y., and Loy, C. C. (2019). Esrgan: Enhanced super-resolution generative adversarial networks. In Leal-Taixé, L. and Roth, S., editors, *Computer Vision – ECCV 2018 Workshops*, pages 63–79.

[209] Wang, Y., Yang, W., Chen, X., Wang, Y., Guo, L., Chau, L.-P., Liu, Z., Qiao, Y., Kot, A. C., and Wen, B. (2024). Sinsr: Diffusion-based image super-resolution in a single step. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 25796–25805.

[210] Wang, Z., Bovik, A., Sheikh, H., and Simoncelli, E. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612.

[211] Wasserman, L. (2006). *All of Nonparametric Statistics*. Springer Science & Business Media.

[212] Weed, J. and Bach, F. (2019). Sharp asymptotic and finite-sample rates of convergence of empirical measures in wasserstein distance.

[213] Wu, H., Köhler, J., and Noé, F. (2020). Stochastic normalizing flows. *arXiv preprint arXiv:2002.06707*.

[214] Xie, S. and Tu, Z. (2015). Holistically-nested edge detection.

[215] Xu, H., Ghosh, G., Song, Y., Zhang, X., Li, C.-Y., Sigal, L., and Wang, Y. (2021). Videoclip: Contrastive pre-training for zero-shot video-text understanding. In *Advances in Neural Information Processing Systems*, volume 34, pages 12471–12484.

[216] Yang, R. and Mandt, S. (2023). Lossy image compression with conditional diffusion models.

[217] Yu, A. and Grauman, K. (2014). Fine-grained visual comparisons with local learning. In *Computer Vision and Pattern Recognition (CVPR)*.

[218] Yu, F., Seff, A., Zhang, Y., Song, S., Funkhouser, T., and Xiao, J. (2015). Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*.

[219] Yu, J. J., Derpanis, K., and Brubaker, M. A. (2020). Wavelet Flow: Fast Training of High Resolution Normalizing Flows. In *NeurIPS*.

[220] Zbontar, J., Jing, L., Misra, I., LeCun, Y., and Deny, S. (2021). Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning (ICML)*. PMLR.

[221] Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. (2018). The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*.

[222] Zhang, X., Zhao, P., Zhang, J., Weng, Y., Yuan, H., Li, Y., and Wang, H. (2021). A unified framework for molecule generation and reaction prediction. *Nature Communications*, 12(1):3117.

[223] Zhao, S., Song, J., and Ermon, S. (2017). Towards deeper understanding of variational autoencoding models.

[224] Zhong, E. D., Bepler, T., Berger, B., and Davis, J. H. (2021). Cryodrgn: reconstruction of heterogeneous cryo-em structures using neural networks. *Nature methods*, 18(2):176–185.

[225] Zhou, D., Wang, Z., Wang, L., Han, J., Dai, X., Shen, Y., and Feng, J. (2022). ibot: Image bert pre-training with online tokenizer. In *International Conference on Learning Representations*.

Appendix A

CAFLOW: Conditional Autoregressive Flows

A.1 Architecture

In this first section of the appendix, we provide more details about the architecture of the unconditional multi-scale flows and the conditional autoregressive multi-scale flows which comprise our modeling framework shown in Figure 3.2.

A.1.1 Unconditional multi-scale flows

The sequence of transformations in each scale are:

1. **Dequantization layer** (only in the first scale). Each pixel of each image channel admits an integer value from 0 to 255. If we do not modify those values, we model a high dimensional distribution in discrete space. However, normalizing flows rely on the rule of change of variables which is naturally defined in continuous space. Therefore, training a normalizing flow on the raw images will result in a model which which will place arbitrarily high likelihood on a few RGB values. This behavior leads to catastrophic destabilization of the training procedure and very poor performance. To remedy this problem, we convert the discrete space to continuous space by dequantizing each pixel value by resampling it from the distribution $q(u_{deq}|u) = \mathscr{U}([u, u+1))$, where $\mathscr{U}([u, u+1))$ is the uniform distribution over the interval $[u, u+1)$. This is called **uniform dequantization** and it is used by the majority of normalizing flow models in the literature. Dequantization effectively represents images as hypercubes. Modeling such sharp borders is challenging for a normalizing flow as it relies on

smooth transformations. For this reason, [76] proposed to learn the dequantization distribution $q(x_{deq}|x)$, where x is the quantized and x_{deq} the dequantized image using a conditional normalizing flow. This is called **variational dequantization**. We noticed that using variational dequantization results in a significant increase in performance. For this reason, we decided not to use variational dequantization in the experiments where we compare against other conditional flows, because this would defeat the purpose of fair comparison. We used variational dequantization in the experiments where we compare against methods which do not rely on normalizing flows.

2. **Squeeze layer**. We squeeze the tensor in the channel dimension effectively halving the spatial resolution: the shape of the tensor is transformed from (B,C,H,W) to $(B,4C,H/2,W/2)$, where B is the batch size, C the number of channels, H the height and W the width of the image.

3. **Two transition steps**. Each transition step consists of an invertible normalization layer (ActNorm) followed by an invertible 1×1 convolution layer. According to [127], the transition step allows the network to learn a linear invertible interpolation between neighboring pixels after the application of the squeeze layer. If the transition steps are not used, the squeeze layer may lead to checkerboard artifacts in the reconstructed image because it is exclusively based on pixel-reordering.

4. **K flow steps**. Each flow step consists of an invertible normalization layer, invertible 1×1 convolution layer and an affine coupling layer in that order. The scale and the bias in the affine coupling layer are computed by a simple convolutional neural network which performs three sequential convolutions with a 3×3, 1×1 and 3×3 convolutional kernels respectively. The number of kernels in each convolutional layer is chosen to be 64.

5. **Split layer**. We split off half of the channels before the squeeze layer of the next scale. Therefore, the next scale transforms only the half part of the tensor. This motivates the modeling of variations in different resolutions and ultimately the hierarchical latent space decomposition of images using multi-scale normalizing flows. Empirically, it has been observed that the first split variables typically encode noise patterns or image details, while the final split variables encode higher level information.

A.1.2 Conditional multi-scale flows

Each scale of a conditional flow F_i^θ contains the following transformations in order:

1. **Squeeze layer**.
2. **Two transition steps**.
3. **M Conditional flow steps**. We adopt the conditional flow step design used by [127]:
   (a) Invertible normalization (ActNorm).
   (b) Invertible 1×1 convolution.
   (c) Affine injector.
   (d) Conditional affine coupling layer.
4. **Split layer**.

We calculate the scale and the bias of the affine transformations in the affine injector and the conditional affine coupling layer using the same convolutional neural network as the one we used for the unconditional flows. The only difference is that we use 32 instead of 64 kernels.

A.2 Hierarchical Representation in Multi-Scale Normalizing Flows

One of the fundamental advantages of employing a multi-scale architecture within normalizing flows lies in its capability to capture latent factors across various levels of abstraction. This phenomenon, where deeper layers tend to encapsulate higher-level information while earlier layers capture finer-grained details, can be attributed to the interplay of two key factors: the depth of computation and the imposed information bottleneck.

The depth of computation factor refers to the fact that in a multi-scale normalizing flow, computation occurs layer by layer, with each layer gradually refining the representation of the input data. When random noise is injected near the flow's output (in the reverse or generative order), most of the computation has already occurred. Consequently, this injected noise primarily influences the remaining local intricacies, such as fine-scale textures or subtle variations in the data. With fewer layers remaining to process, it becomes unlikely for this noise to introduce significant structural alterations or introduce entirely new objects into the scene.

The information bottleneck effect is closely linked to the architectural design of multi-scale normalizing flows, which involves latent space compression in the deeper layers of the normalizing flow.

The presence of this information bottleneck, combined with the depth of computation, compels the network to encode critical information at the deeper layers which don't have enough capacity to represent fine-grained details and simultaneously encode information that is used for the majority of the computation in the flow. Consequently, the network inclines toward prioritizing the encoding of high-level information, such as object identities or overarching scene attributes, within the deeper latent variables.

In essence, the multi-scale architecture's ability to model different levels of abstraction stems from the sequential nature of computation, where each layer refines the representation, and the imposed information bottleneck, which encourages the network to encapsulate essential, high-level details in deeper layers due to their restricted capacity for fine-grained representations.

A.3 Details of experiments

We used the same learning rate scheduling system for all experiments. Initially, we increase the learning rate linearly from 0 to the target learning rate (usually 10^{-3}) in the first 500 iterations. Then, we use the pytorch STEPLR learning rate scheduler with value $\gamma = 0.999$.

When the training curve levels-off (around 70K iterations), we reduce the learning rate by a factor of 10 until final convergence. This reduction provides a final performance boost. We do not repeat that reduction because it only results in overfitting.

Moreover, we used exponential moving average (EMA) with a rate equal to 0.999 in all experiments. We empirically found that EMA consistently provides a small increase in performance. Finally, we clipped the norm of the computed gradients to 1, because it improved training stability with no noticeable compromise on performance.
We adopted the hyperparameter values from the successful approaches in related work, as cited in [193]. Due to the computational intensity of training normalizing flows, hyperparameter tuning becomes computationally prohibitive.

Our primary objective was to demonstrate that our proposed model, which employs autoregressive conditional components, offers improved modelling flexibility compared to non-autoregressive conditional flows like CINN and DualGlow while maintaining computational efficiency. To ensure a fair comparison, we designed our model to have a parameter count that closely aligns with those of the conditional normalizing flow models we referenced. While the GAN-based models also have a similar parameter magnitude, we did not aim for a very close match since our main focus is on contrasting our method with other normalizing flow alternatives.

A.3.1 Image super-resolution

We used 3 scales, 16 flow steps in each level of the conditioning flow R^θ, 32 flow steps in the each level of the conditioned flow T^θ and 12 conditional flow steps in each level of each conditional flow F_i^θ. We set the regularization constant $\lambda = 0.01$. Moreover, we used variational dequantization implemented with a conditional flow of 4 conditional flow steps, because we compared our method against methods which are not based on flows. We trained our model for 4.5 days on a single NVIDIA TESLA P100 GPU on 68K images of the FFHQ dataset with a target learning rate of 10^{-3}. We used 1000 images for validation and 100 images for testing.

A.3.2 Image colorization

We used 3 scales, 24 flow steps in each level of the conditioning flow R^θ, 24 flow steps in the each level of the conditioned flow T^θ and 12 conditional flow steps in each level of each conditional flow F_i^θ. We set the regularization constant $\lambda = 0.01$. We used uniform dequantization as we intended to compare our method with [4], which is a conditional flow. We trained the model on 300K images of the lsun bedroom dataset (this accounts for 10% of

the full dataset) on a single NVIDIA TESLA P100 GPU for 3 days with a target learning rate of 10^{-3}. We used 1000 validation images and 5000 test images.

A.3.3 Image inpainting

We used 3 scales, 30 flow steps in each level of the conditioning flow R^{θ}, 30 flow steps in the each level of the conditioned flow T^{θ} and 16 conditional flow steps in each level of each conditional flow F_i^{θ}. We set the regularization constant $\lambda = 0.05$ and the target learning rate to 10^{-4}, because we faced stability issues with smaller values of λ and greater values of the target learning rate. We trained the model on 195K images of the CelebA dataset for 5 days on a single NVIDIA TESLA P100 GPU. We used 1000 images for validation and 2000 images for testing. We used the same preprocessing as [125] and uniform dequantization.

A.4 Visual results

A.4.1 Image super-resolution

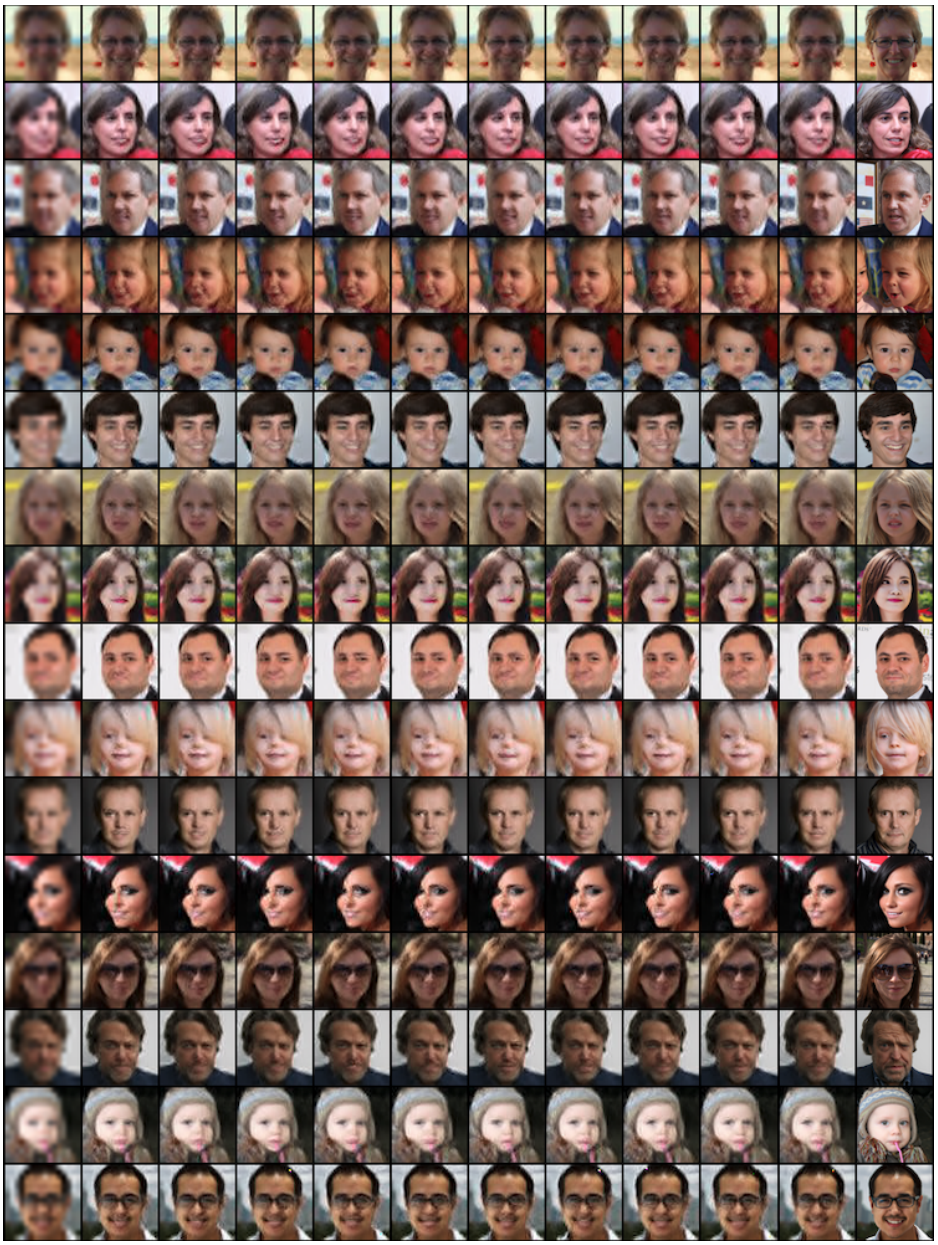


Fig. A.1 Image super-resolution on the FFHQ dataset. Left: LR bicubicly upsampled. Right: HR image. Middle: 10 super-resolved versions in decreasing conditional log-likelihood order from left to right. We sampled 20 super-resolved images for each LR image and we present the 10 images with the highest conditional log-likelihood. We used sampling temperature $\tau = 0.5$.

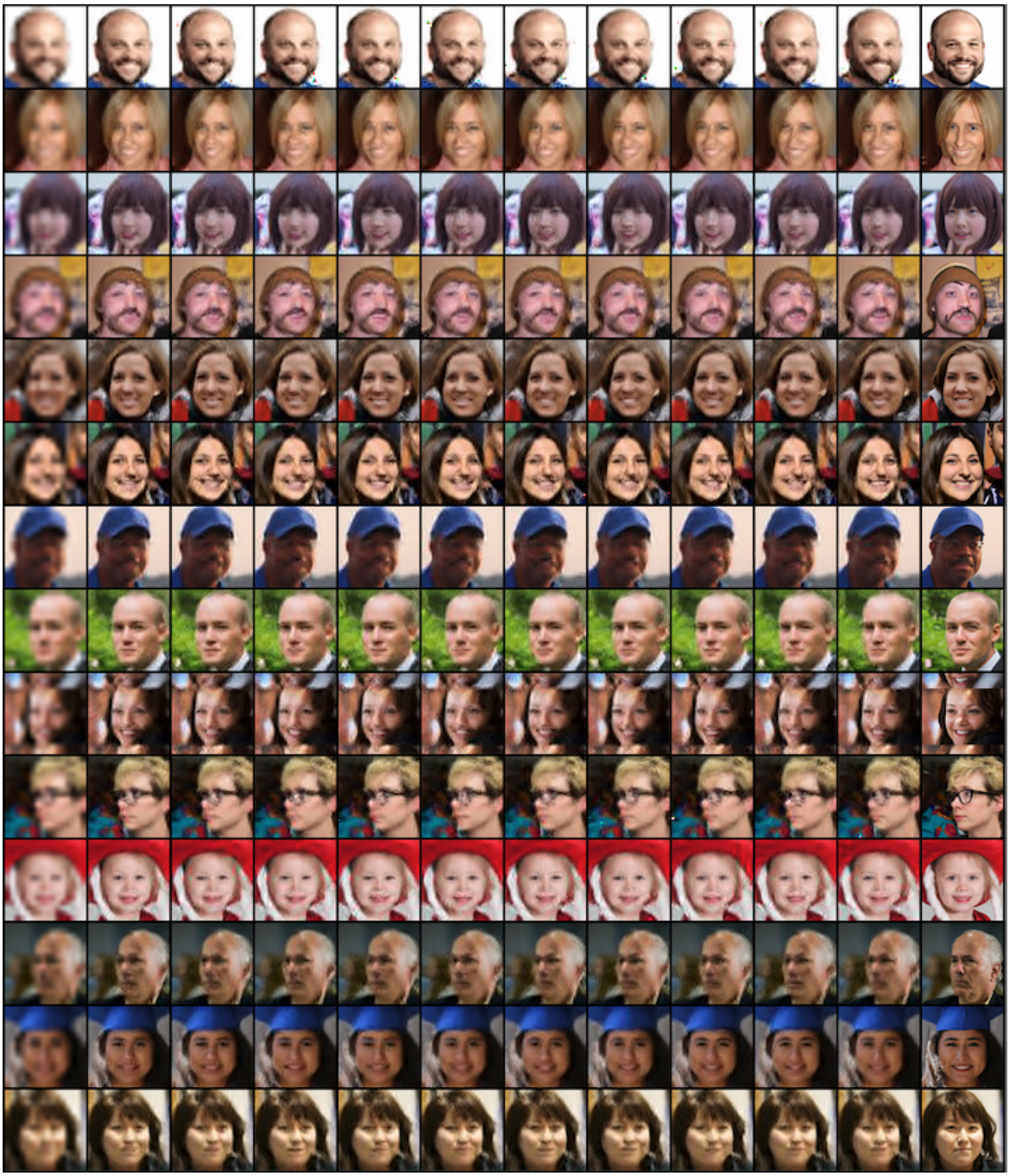


Fig. A.2 Image super-resolution on the FFHQ dataset. Left: LR bicubicly upsampled. Right: HR image. Middle: 10 super-resolved versions in decreasing conditional log-likelihood order from left to right. We sampled 20 super-resolved images for each LR image and we present the 10 images with the highest conditional log-likelihood. We used sampling temperature $\tau = 0.55$.

A.4.2 Image inpainting

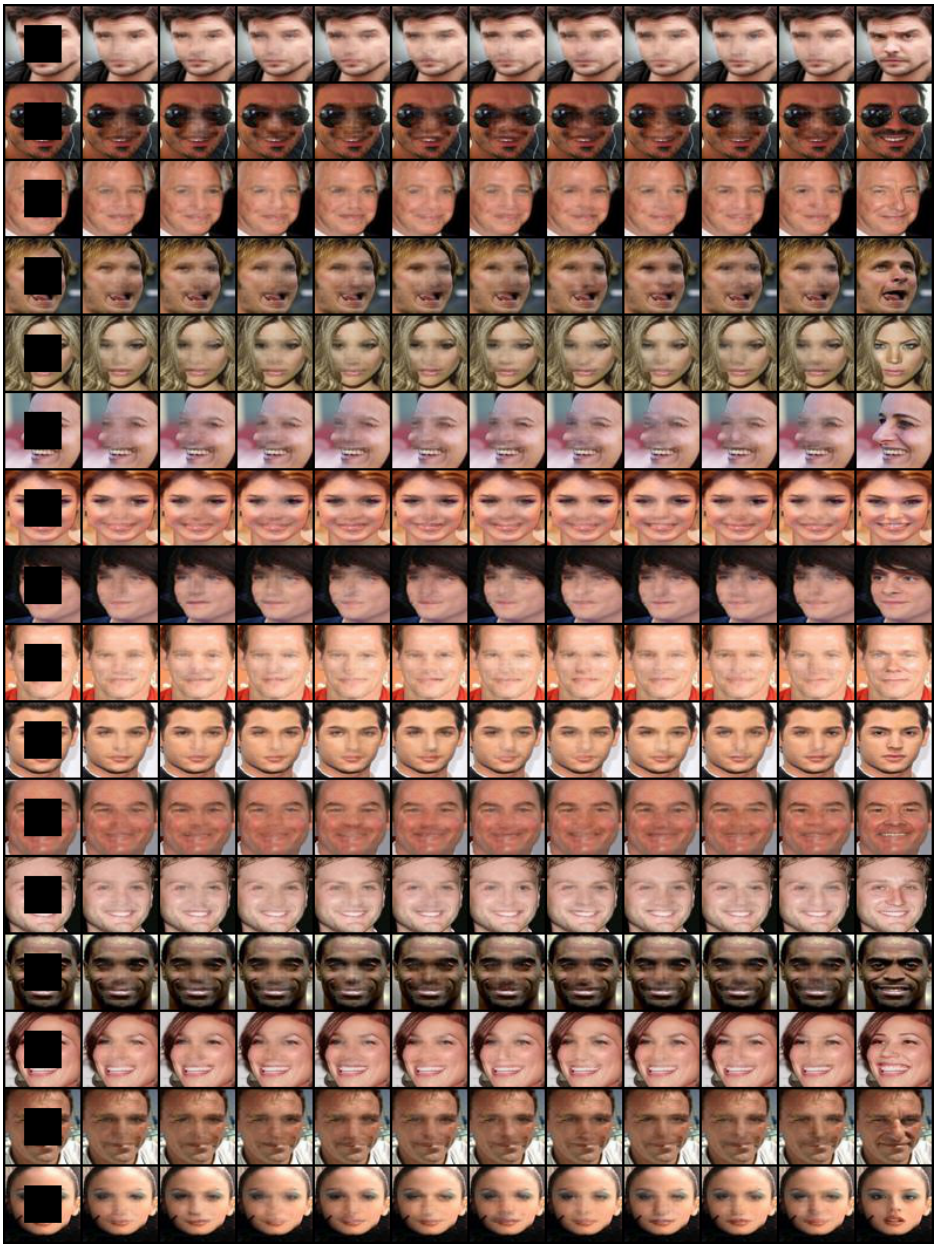


Fig. A.3 Image inpainting on the CelebA dataset. Left: Masked image. Right: Ground truth. Middle: 10 inpainted versions in decreasing conditional log-likelihood order from left to right. We sampled 30 inpainted images for each masked image and we present the 10 images with the highest conditional log-likelihood. We used sampling temperature $\tau = 0.5$.

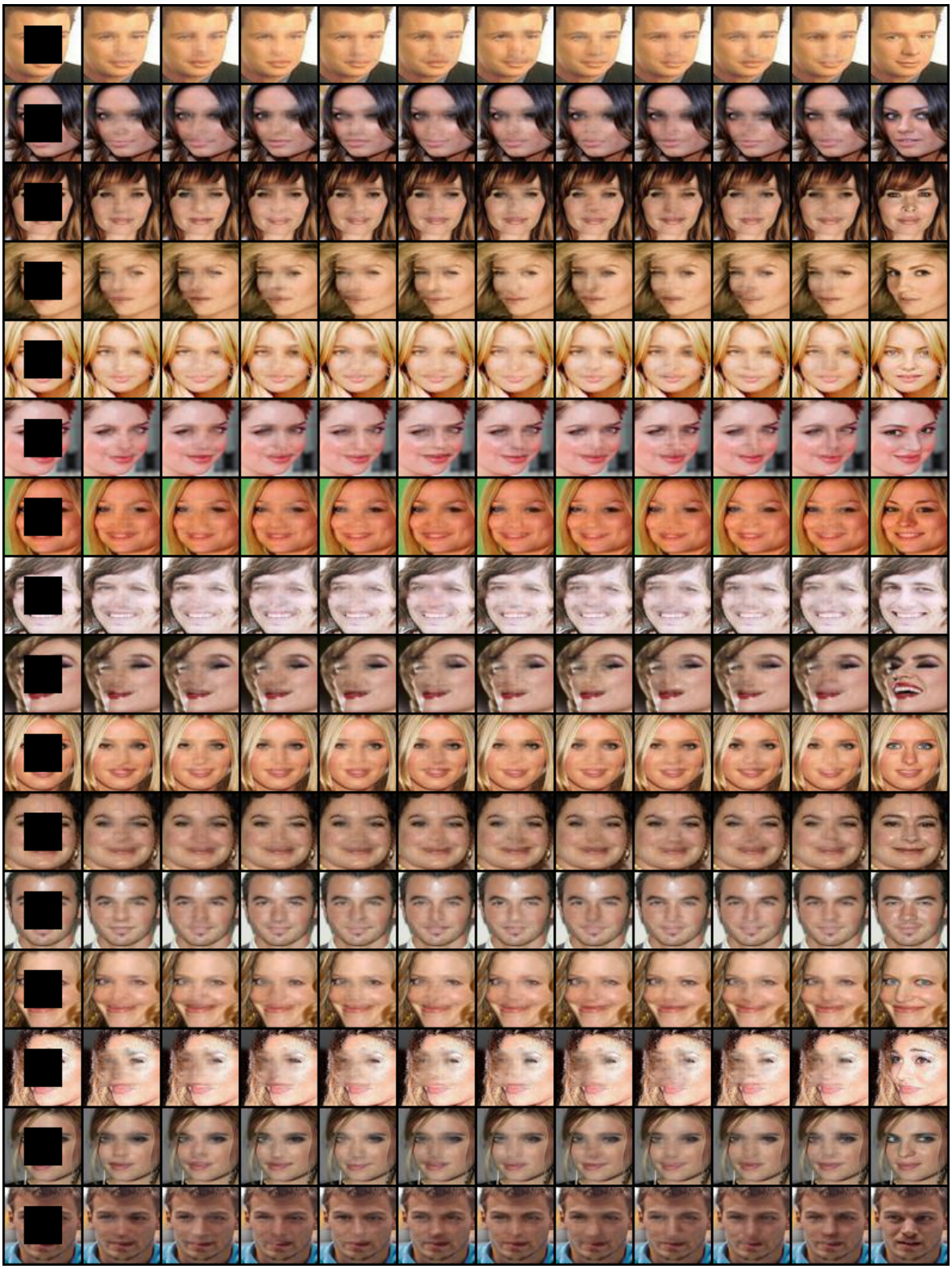


Fig. A.4 Image inpainting on the CelebA dataset. Left: Masked image. Right: Ground truth. Middle: 10 inpainted versions in decreasing conditional log-likelihood order from left to right. We sampled 30 inpainted images for each masked image and we present the 10 images with the highest conditional log-likelihood. We used sampling temperature $\tau = 0.5$.

A.4.3 Image colorization

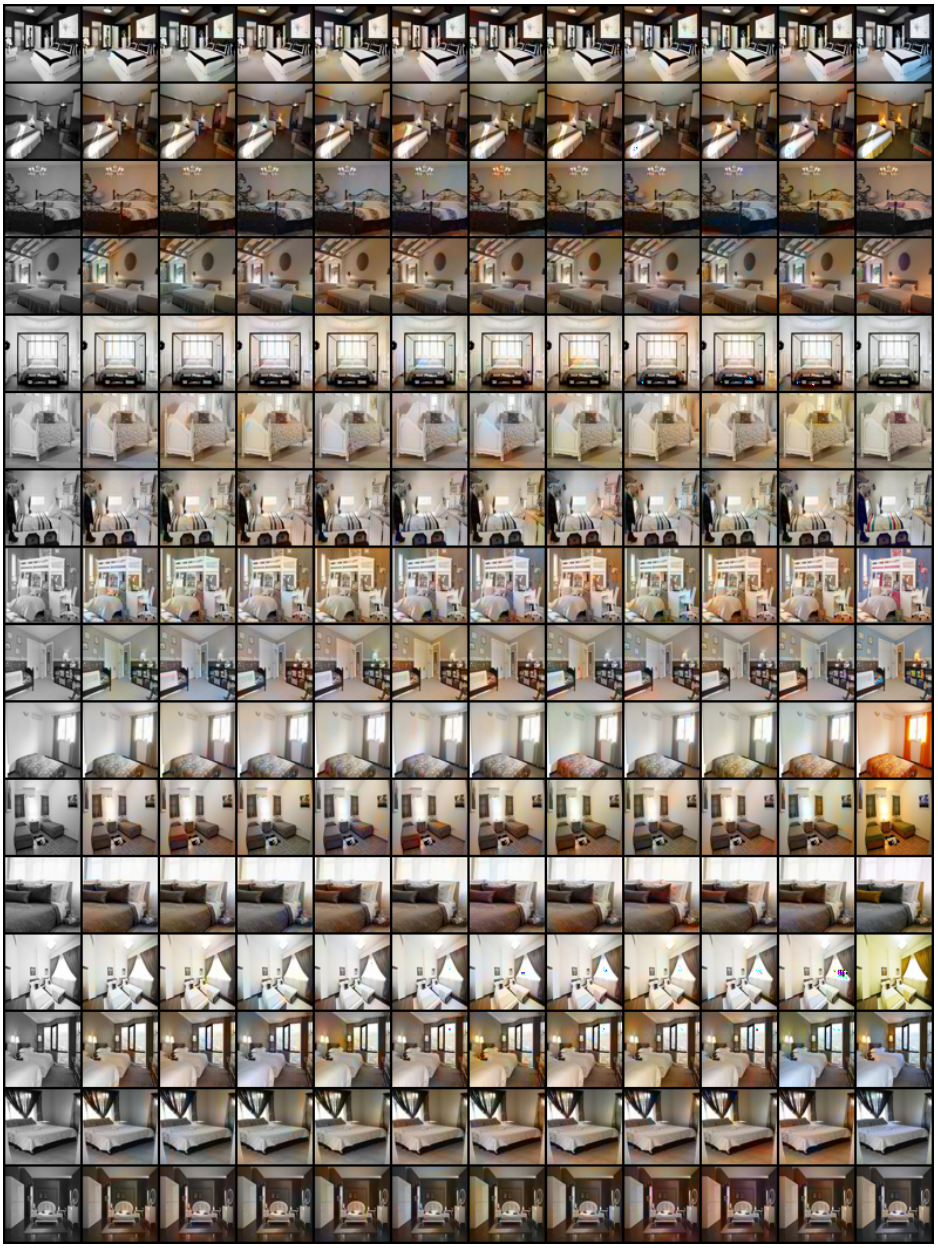


Fig. A.5 Image colorization on the LSUN BEDROOM dataset. Left: Grayscale image. Right: Ground truth. Middle: 10 colorized versions in decreasing conditional log-likelihood order from left to right. We sampled 25 colorized images for each greyscale image and we present the 10 images with the highest conditional log-likelihood. We used sampling temperature $\tau = 0.85$.

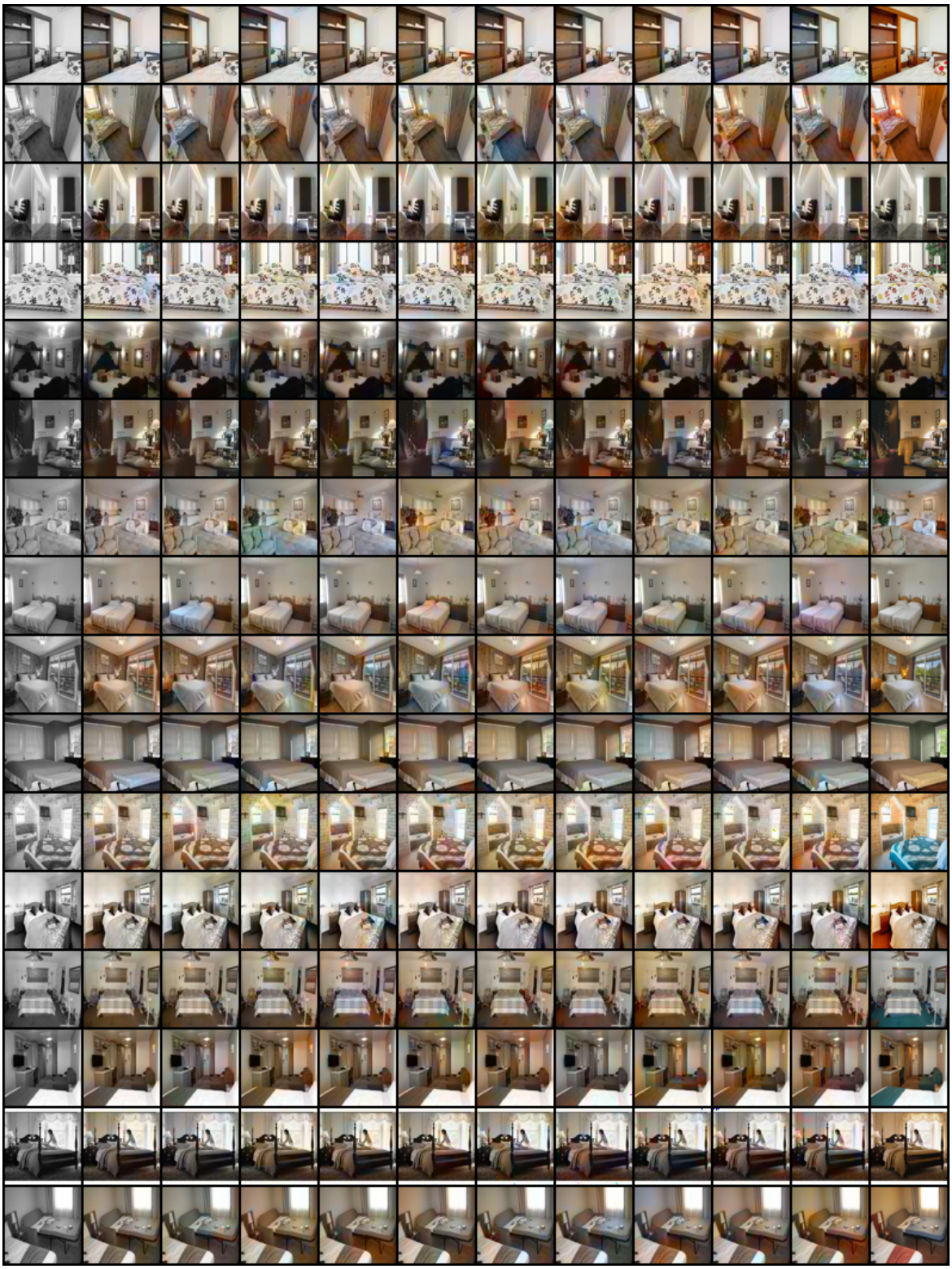


Fig. A.6 Image colorization on the LSUN BEDROOM dataset. Left: Grayscale image. Right: Ground truth. Middle: 10 colorized versions in decreasing conditional log-likelihood order from left to right. We sampled 25 colorized images for each greyscale image and we present the 10 images with the highest conditional log-likelihood. We used sampling temperature $\tau = 0.85$.

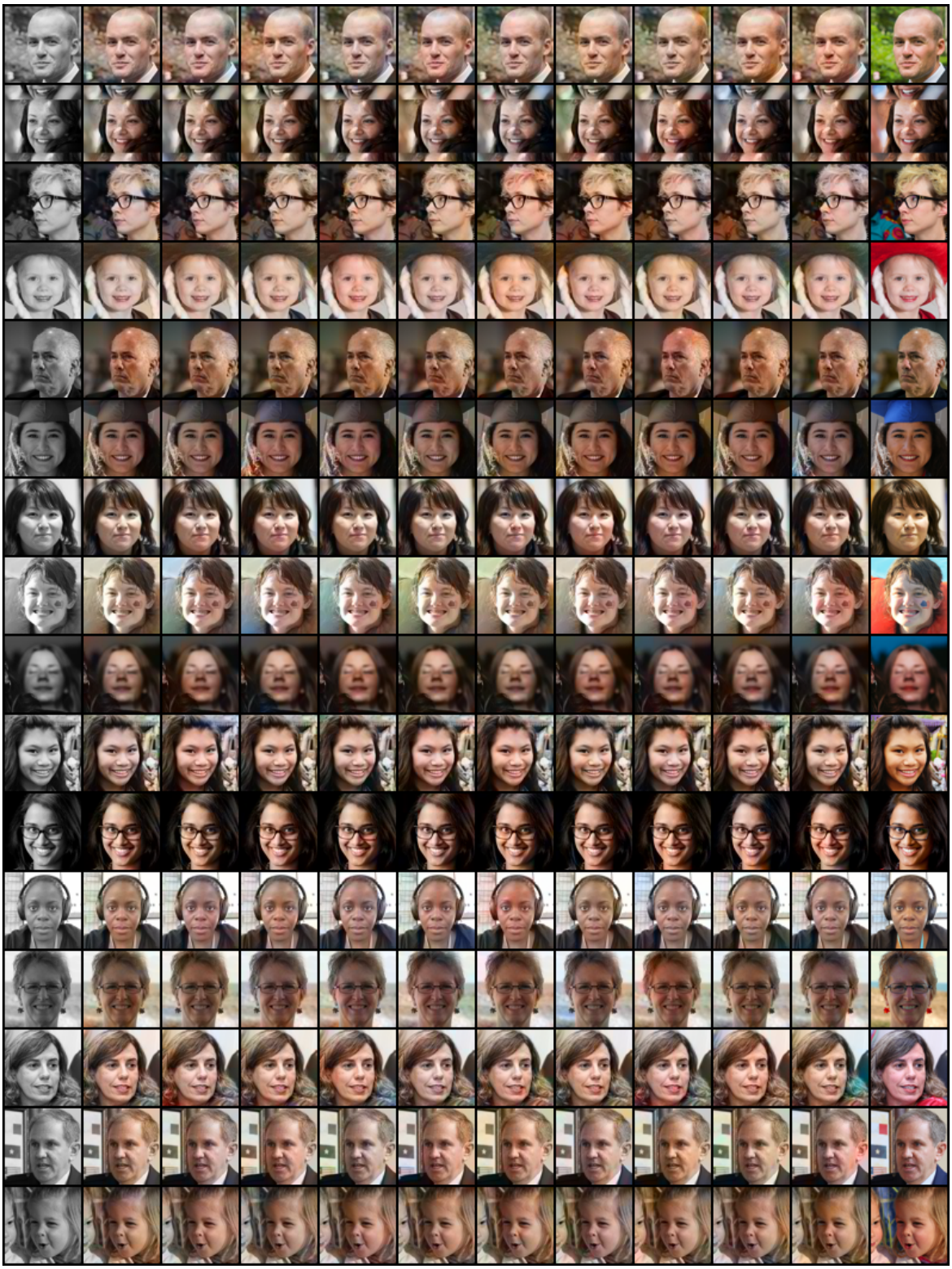


Fig. A.7 Image colorization on the FFHQ dataset. Left: Grayscale image. Right: Ground truth. Middle: 10 colorized versions in decreasing conditional log-likelihood order from left to right. We sampled 25 colorized images for each greyscale image and we present the 10 images with the highest conditional log-likelihood. We used sampling temperature $\tau = 0.7$.

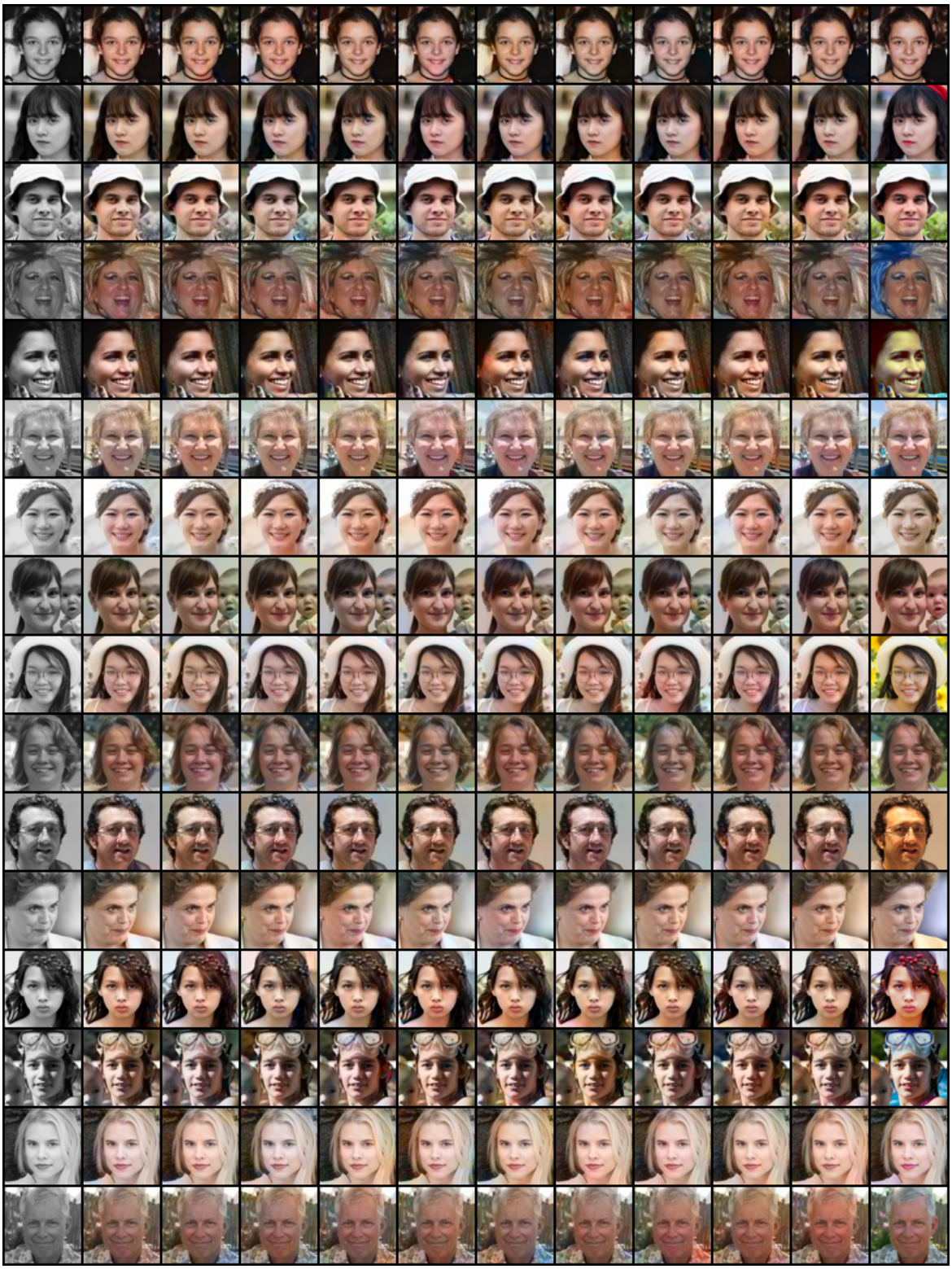


Fig. A.8 Image colorization on the FFHQ dataset. Left: Grayscale image. Right: Ground truth. Middle: 10 colorized versions in decreasing conditional log-likelihood order from left to right. We sampled 25 colorized images for each greyscale image and we present the 10 images with the highest conditional log-likelihood. We used sampling temperature $\tau = 0.7$.

A.4.4 Sketch to image synthesis

Fig. A.9 Sketch to image synthesis on the edges2shoes dataset [86]. Left: Sketch. Right: Ground truth. Middle: 6 samples taken with sampling temperature $\tau = 0.8$.

Appendix B

Non-Uniform Diffusion Models

B.1 Proofs

B.1.1 Equality of minimizers for CDE

Lemma B.1.1. *For a fixed $y \in \mathbb{R}^d$ and $t \in \mathbb{R}$ we have*

$$\begin{aligned}&\mathbb{E}_{\substack{x_0 \sim p(x_0|y) \\ x_t \sim p(x_t|x_0,y)}} [\lambda(t) \|\nabla_{x_t} \ln p(x_t|x_0,y) - s_\theta(x_t,y,t)\|_2^2] \\ =&\mathbb{E}_{x_t \sim p(x_t|y)} [\lambda(t) \|\nabla_{x_t} \ln p(x_t|y) - s_\theta(x_t,y,t)\|_2^2]\end{aligned}$$

Proof. Since y and t are fixed, we may define $\psi(x_t) := s_\theta(x_t,y,t)$, $q(x_0) := p(x_0|y)$ and $q(x_t|x_0) = p(x_t|x_0,y)$. Therefore, by the Tower Law, the statement of the lemma is equivalent to

$$\begin{aligned}&\mathbb{E}_{x_0,x_t \sim q(x_0,x_t)} [\|\nabla_{x_t} \ln q(x_t|x_0) - \psi(x_t)\|_2^2] \\ =&\mathbb{E}_{x_t \sim q(x_t)} [\|\nabla_{x_t} \ln q(x_t) - \psi(x_t)\|_2^2]\end{aligned}$$

Which follows directly from [206, Eq. 11]. □

Theorem 1. *The minimizer of*

$$\frac{1}{2}\mathbb{E}_{\substack{t \sim U(0,T) \\ x_0,y \sim p(x_0,y) \\ x_t \sim p(x_t|x_0)}} [\lambda(t) \|\nabla_{x_t} \ln p(x_t|x_0) - s_\theta(x_t,y,t)\|_2^2]$$

in θ is the same as the minimizer of

$$\frac{1}{2}\mathbb{E}_{\substack{t\sim U(0,T)\\ x_t,y\sim p(x_t,y)}}[\lambda(t)\,\|\nabla_{x_t}\ln p(x_t|y)-s_\theta(x_t,y,t)\|_2^2].$$

Proof. First, notice that x_t is conditionally independent of y given x_0. Therefore, by applying the Tower Law we obtain

$$\begin{aligned}
&\mathbb{E}_{\substack{t\sim U(0,T)\\ x_0,y\sim p(x_0,y)\\ x_t\sim p(x_t|x_0)}}[\lambda(t)\,\|\nabla_{x_t}\ln p(x_t|x_0)-s_\theta(x_t,y,t)\|_2^2]\\
\overset{(1)}{=}&\mathbb{E}_{\substack{t\sim U(0,T)\\ y\sim p(y)\\ x_0\sim p(x_0|y)\\ x_t\sim p(x_t|x_0)}}[\lambda(t)\,\|\nabla_{x_t}\ln p(x_t|x_0)-s_\theta(x_t,y,t)\|_2^2]\\
\overset{(2)}{=}&\mathbb{E}_{\substack{t\sim U(0,T)\\ y\sim p(y)\\ x_0\sim p(x_0|y)\\ x_t\sim p(x_t|x_0,y)}}[\lambda(t)\,\|\nabla_{x_t}\ln p(x_t|x_0,y)-s_\theta(x_t,y,t)\|_2^2]\\
=&\mathbb{E}_{\substack{t\sim U(0,T)\\ y\sim p(y)}}[f(t,y)]=:(*)
\end{aligned}$$

where

$$\begin{aligned}
&f(t,y):=\\
&\mathbb{E}_{\substack{x_0\sim p(x_0|y)\\ x_t\sim p(x_t|x_0,y)}}[\lambda(t)\,\|\nabla_{x_t}\ln p(x_t|x_0,y)-s_\theta(x_t,y,t)\|_2^2].
\end{aligned}$$

Now fix y and t. By Lemma B.1.1, it follows that

$$\begin{aligned}
&f(t,y)\\
=&\mathbb{E}_{\substack{x_0\sim p(x_0|y)\\ x_t\sim p(x_t|x_0,y)}}[\lambda(t)\,\|\nabla_{x_t}\ln p(x_t|x_0,y)-s_\theta(x_t,y,t)\|_2^2]\\
\overset{(3)}{=}&\mathbb{E}_{x_t\sim p(x_t|y)}[\lambda(t)\,\|\nabla_{x_t}\ln p(x_t|y)-s_\theta(x_t,y,t)\|_2^2]
\end{aligned}$$

Since t and y were arbitrary, this is true for all t and y. Therefore, substituting back into $(*)$ we get that

$$\begin{aligned}(*) &= \mathbb{E}_{\substack{t\sim U(0,T)\\ y\sim p(y)\\ x_t\sim p(x_t|y)}}[\lambda(t)\,\|\nabla_{x_t}\ln p(x_t|y) - s_\theta(x_t,y,t)\|_2^2] \\ &\overset{(1)}{=} \mathbb{E}_{\substack{t\sim U(0,T)\\ x_t,y\sim p(x_t,y)}}[\lambda(t)\,\|\nabla_{x_t}\ln p(x_t|y) - s_\theta(x_t,y,t)\|_2^2].\end{aligned}$$

(1) Tower Law, (2) Conditional independence of x_t and y given x_0, (3) Lemma B.1.1. □

B.1.2 Consistency of CDE

In order to prove the consistency, in this subsection we make the following assumptions:

Assumption B.1.3. The space of parameters Θ and the data space $\mathscr{X}$ are compact.

Assumption B.1.4. There exists a unique $\theta^* \in \Theta$ such that $s_{\theta^*}(x,y,t) = \nabla_{x_t}\ln p(x,y,t)$.

First we state some technical, but well-known lemmas, which will be useful in proving our consistency result.

Lemma B.1.5 (Uniform law of large numbers)**.** *[135, Lemma 2.4]*
Let z_i be i.i.d from a distribution $q(z)$ and suppose that:

- *Θ is compact.*
- *$f(z,\theta)$ is continuous for all $\theta \in \Theta$ and almost all z.*
- *$f(\cdot,\theta)$ is a measurable function of z for each θ.*
- *There exists $d : \mathscr{Z} \to \mathbb{R}$ such that $\mathbb{E}[d(z)] < \infty$ and $\|f(z,\theta)\| \le d(z)$ for each θ.*

Then $\mathbb{E}_z[f(z,\theta)]$ is continuous in θ, and $\frac{1}{n}\sum_{i=1}^n f(z_i,\theta)$ converges to $\mathbb{E}_z[f(z,\theta)]$ uniformly in probability, i.e.:

$$\sup_\theta \left\| \frac{1}{n}\sum_{i=1}^n f(z_i,\theta) - \mathbb{E}_z[f(z,\theta)] \right\| \xrightarrow{P} 0$$

Lemma B.1.6 (Consistency of extremum estimators)**.** *[135, Theorem 2.1]*
Let Θ be compact and consider a family of functions $\mathscr{L}^{(n)} : \Theta \to \mathbb{R}$. Moreover, suppose there exists a function $\mathscr{L} : \Theta \to \mathbb{R}$ such that

- $\mathscr{L}(\theta)$ *is uniquely minimized at* θ^*.
- $\mathscr{L}(\theta)$ *is continuous.*
- $\mathscr{L}^{(n)}(\theta)$ *converges uniformly in probability to* $\mathscr{L}(\theta)$.

Then

$$\theta_n^* := \arg\min_{\theta\in\Theta} \mathscr{L}^{(n)}(\theta).$$

Corollary B.1.7. *Let* θ_n^* *be a minimizer of a n-sample Monte Carlo approximation of*

$$\frac{1}{2}\mathbb{E}_{\substack{t\sim U(0,T)\\ x_0,y\sim p(x_0,y)\\ x_t\sim p(x_t|x_0)}}[\lambda(t)\,\|\nabla_{x_t}\ln p(x_t|x_0) - s_\theta(x_t,y,t)\|_2^2].$$

Then under assumptions B.1.3 and B.1.4, the conditional denoising estimator $s_{\theta_n^*}(x,y,t)$ *is a consistent estimator of the conditional score* $\nabla_{x_t}\ln p(x_t|y)$*, i.e.*

$$s_{\theta_n^*}(x,y,t) \xrightarrow{P} \nabla_{x_t}\ln p(x_t|y),$$

as the number of Monte Carlo samples n approaches infinity.

Proof. By conditional independence and the Tower Law, we get

$$\begin{aligned}
&\mathbb{E}_{\substack{t\sim U(0,T)\\ x_0,y\sim p(x_0,y)\\ x_t\sim p(x_t|x_0)}}[\lambda(t)\,\|\nabla_{x_t}\ln p(x_t|x_0) - s_\theta(x_t,y,t)\|_2^2]\\
=&\mathbb{E}_{\substack{t\sim U(0,T)\\ x_0,y\sim p(x_0,y)\\ x_t\sim p(x_t|x_0,y)}}[\lambda(t)\,\|\nabla_{x_t}\ln p(x_t|x_0) - s_\theta(x_t,y,t)\|_2^2]\\
=&\mathbb{E}_{\substack{t\sim U(0,T)\\ x_0,,x_t,y\sim p(x_0,x_t,y)}}[\lambda(t)\,\|\nabla_{x_t}\ln p(x_t|x_0) - s_\theta(x_t,y,t)\|_2^2].
\end{aligned}$$

Let $z=(t,x_0,x_t,y)$ and denote by $q(z) := p(t,x_0,x_t,y)$ the joint distribution. Moreover, define $f(z,\theta) := \lambda(t)\,\|\nabla_{x_t}\ln p(x_t|x_0) - s_\theta(x_t,y,t)\|_2^2$. Since $t\sim U(0,T)$ is independent of $(x_0,x_t,y)\sim p(x_0,x_t,y)$, the above is equal to

$$\mathbb{E}_{z\sim q(z)}[f(z,\theta)]$$

Therefore by Lemma B.1.5, the Monte Carlo approximation of 4.12: $\mathscr{L}^{(n)}(\theta) = \frac{1}{n}\sum_{i=1}^n f(z_i,\theta)$ converges uniformly in probability to $\mathscr{L}(\theta) = \mathbb{E}_{z\sim q(z)}[f(z,\theta)]$. Let θ^* be the minimizer of $\mathscr{L}(\theta)$, by Lemma B.1.6 we get that $\theta_n^* \xrightarrow{P} \theta^*$. Finally by Theorem 4.3.2, θ^* is also a

minimizer of the Fisher divergence between $s_{\theta^*}(x_t, y, t)$ and $\nabla_{x_t} \ln p(x_t|y)$ and by Assumption B.1.4 this implies that $s_{\theta^*}(x_t, y, t) = \nabla_{x_t} \ln p(x_t|y)$. Hence $s_{\theta_n^*}(x, y, t) \xrightarrow{P} \nabla_{x_t} \ln p(x_t|y)$. □

B.1.3 Likelihood weighting for multi-speed and multi-sde models

In this section we derive the likelihood weighting for multi-sde models (Theorem 4.3.4). First using the framework in [186, Appendix A] we present the Anderson's theorem for multi-dimensional SDEs with non-homogeneous covariance matrix (without assuming $\Sigma(t) \neq \sigma(t)I$) and generalize the main result of [181] to this setting. Then, we cast the problem of multi-speed and multi-sde diffusion as a special case of multi-dimensional diffusion with a particular covariance matrix $\Sigma(t)$ and thus obtain the likelihood weighting for multi-sde models (Theorem 4.3.4).

Consider an Ito's SDE

$$dx = \mu(x,t)dt + \Sigma(t)dw$$

where $\mu : \mathbb{R}^{n_x} \times [0,T] \to \mathbb{R}^{n_x}$ and $\Sigma : [0,T] \to \mathbb{R}^{n_x \times n_x}$ is a time-dependent positive-definite matrix. By multi-dimensional Anderson's Theorem [2] the corresponding reverse time SDE is given by

$$\begin{aligned} dx &= \tilde{\mu}(x,t)dt + \Sigma(t)dw \\ \text{where } \tilde{\mu}(x,t) &:= \mu(x,t) - \Sigma(t)^2 \nabla_x \ln p_{X_t}(x). \end{aligned} \tag{B.1}$$

If we train a score-based diffusion model to approximate $\nabla_x \ln p_{X_t}(x)$ with a neural network $s_\theta(x,t)$ we will obtain the following approximate reverse-time sde

$$\begin{aligned} dx &= \tilde{\mu}_\theta(x,t)dt + \Sigma(t)dw \\ \text{where } \tilde{\mu}_\theta(x,t) &:= \mu(x,t) - \Sigma(t)^2 s_\theta(x,t) \end{aligned} \tag{B.2}$$

Now we generalize [181, Theorem 1] to multi-dimensional setting.

Theorem B.1.8. *Let $p(x_t)$ and $p_\theta(x_t)$ denote marginal distributions of B.1 and B.2 respectively. Then under regularity assumptions of [181, Theorem 1] we have that*

$$KL(p(x_0)|p_\theta(x_0)) \leq KL(p(x_T)|\pi(x_T)) + \frac{1}{2}\mathbb{E}_{\substack{t\sim U(0,T) \\ x_t \sim p(x_t)}}[v^T \Sigma(t)^2 v], \tag{B.3}$$

where $v = \nabla_{x_t} \ln p(x_t) - s_\theta(x_t, t)$.

Proof. We proceed in close analogy to the proof of [181, Theorem 1] but we use a more general diffusion matrix $\Sigma(t)$. Let P be the law of the true reverse-time sde and let P_θ be the law of the approximate reverse-time sde. Then by [128, Theorem 2.4] (generalized chain rule for KL divergence) we have

$$KL(P|P_\theta) = KL(p(x_0)|p_\theta(x_0)) + \mathbb{E}_{z\sim p(x_0)}[KL(P(\cdot|x_0 = z)|P_\theta(\cdot|x_0 = z))]. \tag{B.4}$$

Since $\mathbb{E}_{z\sim p(x_0)}[KL(P(\cdot|x_0 = z)|P_\theta(\cdot|x_0 = z))] \geq 0$, this implies

$$KL(p(x_0)|p_\theta(x_0)) \leq KL(P|P_\theta)$$

Using the fact that $p_\theta(x_T) = \pi$ and applying [128, Theorem 2.4] again, we obtain

$$KL(P|P_\theta) = KL(p(x_T)|\pi) + \mathbb{E}_{z\sim p(x_T)}[KL(P(\cdot|x_T = z)|P_\theta(\cdot|x_T = z))]. \tag{B.5}$$

Let $P^z := P(\cdot|x_T = z)$ and $P^z_\theta := P_\theta(\cdot|x_T = z)$

$$\mathbb{E}_{z\sim p(x_T)}[KL(P(\cdot|x_T = z)|P_\theta(\cdot|x_T = z))] = -\mathbb{E}_{z\sim p(x_T)}\left[\mathbb{E}_{P^z}\left[\ln \frac{dP^z_\theta}{dP^z}\right]\right]. \tag{B.6}$$

Using Girsanov Theorem [139, Theorem 8.6.5] and the fact that $\Sigma(t)$ is symmetric and invertible

$$\mathbb{E}_{z\sim p(x_T)}\left[\mathbb{E}_{P^z}\left[\int_0^T \Sigma(t)v(x_t,t)dw_t + \frac{1}{2}\int_0^T v(x_t,t)^T\Sigma(t)^2 v(x_t,t)dt\right]\right]. \tag{B.7}$$

where $v(x_t,t) = \nabla_{x_t} \ln p(x_t) - s_\theta(x_t,t)$. Since $\int_0^T \Sigma(t)v(x_t,t)dw_t$ is a martingale (Ito's integral wrt Brownian motion)

$$\begin{aligned}
&= \frac{1}{2}\mathbb{E}_{z\sim p(x_T)}\left[\mathbb{E}_{P^z}\left[\int_0^T v(x_t,t)^T\Sigma(t)^2 v(x_t,t)dt\right]\right] \\
&= \frac{1}{2}\int_0^T \mathbb{E}_{x\sim p(x_t)}[v(x_t,t)^T\Sigma(t)^2 v(x_t,t)] \\
&= \frac{1}{2}\mathbb{E}_{\substack{t\sim U(0,T)\\ x_t\sim p(x_t)}}[v(x_t,t)^T\Sigma(t)^2 v(x_t,t)].
\end{aligned}$$

□

Multi-sde and multi-speed diffusion

Now we consider again the multi-speed and the more general multi-sde diffusion frameworks from sections 4.3.3 and 4.3.4. Suppose that we have two tensors x and y which diffuse according to different SDEs

$$dx = \mu^x(x,t)dt + \sigma^x(t)dw$$
$$dy = \mu^y(y,t)dt + \sigma^y(t)dw$$

We may cast this system of two SDEs, as a single SDE

$$dz = \mu^z(z,t)dt + \Sigma_z(t)dw$$

where $z = (x,y)$, $\mu^z(z,t) = (\mu^x(x,t), \mu^y(x,t))$ and

$$\Sigma_z(t) = \begin{cases} \sigma^x(t), \text{ if } i = j,\ i \leq n_x \\ \sigma^y(t), \text{ if } i = j,\ n_x < i \leq n_y \\ 0, \text{ otherwise} \end{cases} .$$

If we train a score-based diffusion model for $z_t = (x_t, y_t)$, then by Theorem B.1.8

$$KL(p(z_0)|p_\theta(z_0)) \leq C_1 + \frac{1}{2}\mathbb{E}_{\substack{t\sim U(0,T) \\ z_t\sim p(z_t)}}[v^T \Sigma_z(t)^2 v],$$

where $C_1 := KL(p(x_T)|\pi(x_T))$ does not depend on θ. Because Λ_{MLE} (from Theorem 4.3.4) is equal to $\Sigma_z(t)^2$, we may rewrite the above as

$$KL(p(z_0)|p_\theta(z_0)) \leq C_1 + \frac{1}{2}\mathbb{E}_{\substack{t\sim U(0,T) \\ z_t\sim p(z_t)}}[v^T \Lambda_{MLE}(t)^2 v],$$

and since by denoising score matching [206]

$$\mathbb{E}_{\substack{t\sim U(0,T) \\ z_t\sim p(z_t)}}\left[v^T \Lambda_{MLE}(t) v\right] = \mathbb{E}_{\substack{t\sim U(0,T) \\ z_0\sim p_0(z_0) \\ z_t\sim p(z_t|z_0)}}\left[v^T \Lambda_{MLE}(t) v\right] + C_2. \tag{B.8}$$

where C_2 is another term constant in θ. We conclude that

$$KL(p(z_0)|p_\theta(z_0)) \leq \frac{1}{2}\mathbb{E}_{\substack{t\sim U(0,T) \\ z_0\sim p_0(z_0) \\ z_t\sim p(z_t|z_0)}}[v^T \Lambda_{MLE}(t) v] + C_3$$

where $C_3 := C_1 + C_2$. Now recall that the term on the RHS is exactly the training objective of a multi-sde score-based diffusion model with likelihood weighting

$$\mathscr{L}(\theta) := \frac{1}{2}\mathbb{E}_{\substack{t\sim U(0,T)\\ z_0\sim p_0(z_0)\\ z_t\sim p(z_t|z_0)}}[v^T \Lambda_{MLE}(t) v].$$

Therefore

$$KL(p(z_0)|p_\theta(z_0)) \leq \mathscr{L}(\theta) + C_3.$$

Finally, since $KL(p(z_0)|p_\theta(z_0)) = \mathbb{E}_{(x,y)\sim p(x,y)}[\ln p(x,y)] - \mathbb{E}_{(x,y)\sim p(x,y)}[\ln p_\theta(x,y)]$, we have

$$-\mathbb{E}_{(x,y)\sim p(x,y)}[\ln p_\theta(x,y)] \leq \mathscr{L}(\theta) + C$$

where $C := C_3 - \mathbb{E}_{(x,y)\sim p(x,y)}[\ln p(x,y)]$ is independent of θ. Thus the Theorem 4.3.4 is established.

B.1.4 Mean square approximation error

Assumption B.1.9. $p(x,y) \in C^2(\mathscr{X})$

Assumption B.1.10. $p(x,y) > 0$ for all x, y.

Assumption B.1.11. The data space $\mathscr{X}$ is compact.

Lemma B.1.12. *Under assumptions B.1.9 and B.1.11 we have*

$$p_{Y_t|X_t}(y_t|x_t) = (p_{Y|X_t}(\cdot|x_t) * \varphi_\sigma)(y_t)$$
$$\partial_{x_t} p_{Y_t|X_t}(y_t|x_t) = (\partial_{x_t} p_{Y|X_t}(\cdot|x_t) * \varphi_\sigma)(y_t)$$

Proof. For this proof, we drop our convention of denoting the probability distribution of a random variable via the name of its density's argument.

$$\begin{aligned} p_{Y_t|X_t}(y_t|x_t) &= \int p_{Y,Y_t|X_t}(y, y_t|x_t)dy \\ &= \int p_{Y|X_t}(y|x_t) p_{Y_t|Y,X_t}(y_t|y,x_t)dt \\ &= \int p_{Y|X_t}(y|x_t) p_{Y_t|Y}(y_t|y)dy \end{aligned}$$

Since $Y_t|Y$ has normal distribution with mean y and variance $\sigma^y(t)^2$:

$$\begin{aligned}&= \int p_{Y|X_t}(y|x_t)\varphi_\sigma(y_t - y)dy \\ &= (p_{Y|X_t}(\cdot|x_t) * \varphi_\sigma)(y_t)\end{aligned}$$

where φ_σ is a Gaussian kernel with variance $\sigma^y(t)^2$. Moreover, under the assumptions of the lemma we can exchange the differentiaion and integration. Therefore

$$\begin{aligned}\partial_{x_t} p_{Y_t|X_t}(y_t|x_t) &= \partial_{x_t} \int p_{Y|X_t}(y|x_t)\varphi_\sigma(y_t - y)dy \\ &= \int \partial_{x_t} p_{Y|X_t}(y|x_t)\varphi_\sigma(y_t - y)dy \\ &= (\partial_{x_t} p_{Y|X_t}(\cdot|x_t) * \varphi_\sigma)(y_t)\end{aligned}$$

□

Lemma B.1.13. *Let f be a C^1-function on a compact domain $\mathscr{X}$ and let φ_σ be a Gaussian kernel with variance σ^2. Then there exists a function $E : \mathbb{R} \to \mathbb{R}$, which is monotonically decreasing to zero, such that*

$$\|(f * \varphi_\sigma) - f\|_\infty \leq E(1/\sigma).$$

Proof.

$$\begin{aligned}&|(f * \varphi_\sigma)(y) - f(y)| \\ &= \left| \int f(z)\varphi_\sigma(z-y)dz - \int f(y)\varphi_\sigma(z-y)dz \right| \\ &\leq \int |f(z) - f(y)|\varphi_\sigma(z-y)dz\end{aligned}$$

Since f is a C^1 function on a compact domain, it is Lipschitz and bounded (in absolute value) by some constant M. Fix $\varepsilon > 0$, and let L denote the Lipschitz constant of f. We have that $|f(z) - f(y)| < \varepsilon$ whenever $\|z - y\| < \varepsilon/L$. Let $B_y(\varepsilon/L) := \{z \in \mathscr{X} : \|z - y\| < \varepsilon/L\}$ be a

ball of radius ε/L around y. Then

$$\begin{aligned}
&\int |f(z)-f(y)|\varphi_\sigma(z-y)dz \\
&= \int_{B_y(\varepsilon/L)} |f(z)-f(y)|\varphi_\sigma(z-y)dz \\
&\qquad + \int_{\mathscr{X}\setminus B_y(\varepsilon/L)} |f(z)-f(y)|\varphi_\sigma(z-y)dz \\
&\leq \varepsilon + \int_{\mathscr{X}\setminus B_y(\varepsilon/L)} 2M\varphi_\sigma(z-y)dz \\
&= \varepsilon + 2MP\left(|Z_\sigma| > \frac{\varepsilon}{L}\right)
\end{aligned}$$

where Z_σ is a normally-distributed random variable with mean zero and variance σ^2. By the Chernoff bound, we have

$$\leq \varepsilon + 4M\exp\left(-\frac{\varepsilon^2}{2L^2\sigma^2}\right).$$

Define $E_\varepsilon(1/\sigma) := \varepsilon + 4M\exp\left(-\frac{\varepsilon^2}{2L^2\sigma^2}\right)$. Observe that $E_\varepsilon : \mathbb{R}_+ \to \mathbb{R}$ is monotonically decreasing to ε. Moreover

$$\|(f*\varphi_\sigma) - f\|_\infty \leq E_\varepsilon(1/\sigma).$$

Now let $A := [0,1]$ and define

$$E(1/\sigma) := \min_{\varepsilon\in A} E_\varepsilon(1/\sigma).$$

Notice that the above minimum is achieved, since A is compact and for a fixed σ, the function $\varepsilon \mapsto E_\varepsilon(1/\sigma)$ is continuous.

We will prove that E is a monotonically decreasing to zero and upper-bounds $\|(f*\varphi_\sigma) - f\|_\infty$. Firstly, it is clear that $E(x) \to 0$ as $x \to \infty$, since for all $\varepsilon \in A$ we have $\lim_{x\to\infty} E(x) \leq \lim_{x\to\infty} E_\varepsilon(x) = \varepsilon$. Secondly, suppose $a < b$, and let ε_a be such that $E(a) = E_{\varepsilon_a}(a)$. Then

$$E(b) = \inf_{\varepsilon\in A} E_\varepsilon(b) \leq E_{\varepsilon_a}(b) < E_{\varepsilon_a}(a) = E(a).$$

Therefore E is monotonically decreasing. Finally since for all $\varepsilon > 0$

$$\|(f*\varphi_\sigma) - f\|_\infty \leq E_\varepsilon(1/\sigma).$$

Taking minimum over $\varepsilon \in A$ on both sides we obtain

$$\|(f * \varphi_\sigma) - f\|_\infty \leq E(1/\sigma).$$

□

Lemma B.1.14. *Let f be a C^1 function on a compact domain and let Z be a random variable with mean μ and variance σ^2. Then*

$$\mathbb{E}_Z[(f(\mu) - f(Z))^2] \leq L^2\sigma^2$$

where L denotes the Lipschitz constant of f.

Proof. Since f is a C^1 function on a compact domain it is Lipschitz with some Lipschitz constant L. Therefore

$$\mathbb{E}_Z[(f(\mu) - f(Z))^2] \leq L^2\mathbb{E}_Z[(\mu - Z)^2] \leq L^2\sigma^2$$

□

Theorem 3. *Fix t, x_t and y. Then under Assumptions B.1.9, B.1.10 and B.1.11, there exists a function $E : \mathbb{R} \to \mathbb{R}$ which is monotonically decreasing to zero, such that*

$$\mathbb{E}_{y_t \sim p(y_t|y)}\left[\|\nabla_{x_t} \ln p(x_t|y_t) - \nabla_{x_t} \ln p(x_t|y)\|_2^2\right] \leq E\left(\frac{1}{\sigma^y(t)}\right).$$

Proof. For this proof, we drop our convention of denoting the probability distribution of a random variable via the name of its density's argument.

$$\left\|\nabla_{x_t} \ln p_{X_t|Y_t}(x_t|y_t) - \nabla_{x_t} \ln p_{X_t|Y}(x_t|y)\right\|_2^2 = \sum_{i=1}^{n_x} \left(\partial^i_{x_t} \ln p_{X_t|Y_t}(x_t|y_t) - \partial^i_{x_t} \ln p_{X_t|Y}(x_t|y)\right)^2$$

Therefore it is sufficient to prove the theorem in each dimension separately. Hence, without loss of generality, we may assume that $x_t \in \mathbb{R}$ and show

$$\mathbb{E}_{y_t \sim p(y_t|y)}\left[\left(\partial_{x_t} \ln p_{X_t|Y_t}(x_t|y_t) - \partial_{x_t} \ln p_{X_t|Y}(x_t|y)\right)^2\right] \leq E\left(\frac{1}{\sigma^y(t)}\right).$$

By Bayes's rule we have

$$\begin{aligned}
\partial_{x_t} \ln p_{X_t|Y_t}(x_t|y_t) &= \partial_{x_t} \ln p_{Y_t|X_t}(y_t|x_t) + \partial_{x_t} \ln p_{X_t}(x_t) \\
\partial_{x_t} \ln p_{X_t|Y}(x_t|y) &= \partial_{x_t} \ln p_{Y|X_t}(y|x_t) + \partial_{x_t} \ln p_{X_t}(x_t).
\end{aligned}$$

Therefore,

$$\left(\partial_{x_t} \ln p_{X_t|Y_t}(x_t|y_t) - \partial_{x_t} \ln p_{X_t|Y}(x_t|y)\right)^2 = \left(\partial_{x_t} \ln p_{Y_t|X_t}(y_t|x_t) - \partial_{x_t} \ln p_{Y|X_t}(y|x_t)\right)^2.$$

To unclutter the notation, let $p(y|x) := p_{Y|X_t}(y|x)$ and $p_\sigma(y|x) := p_{Y_t|X_t}(y|x)$. Applying this notation:

$$\begin{aligned}&\mathbb{E}_{y_t\sim p(y_t|y)}[(\partial_{x_t} \ln p_{Y_t|X_t}(y_t|x_t) - \partial_{x_t} \ln p_{Y|X_t}(y|x_t))^2] \\ &\quad = \mathbb{E}_{y_t\sim p(y_t|y)}[(\partial_{x_t} \ln p_\sigma(y_t|x_t) - \partial_{x_t} \ln p(y|x_t))^2]\end{aligned}$$

Adding and subtracting $\partial_{x_t} \ln p(y_t|x_t)$ and using the triangle inequality:

$$\begin{aligned}&\leq \mathbb{E}_{y_t\sim p(y_t|y)}[(\partial_{x_t} \ln p_\sigma(y_t|x_t) - \partial_{x_t} \ln p(y_t|x_t))^2] \\ &\quad + \mathbb{E}_{y_t\sim p(y_t|y)}[(\partial_{x_t} \ln p(y_t|x_t) - \partial_{x_t} \ln p(y|x_t))^2]\end{aligned}$$

We may bound the expectation by the supremum norm

$$\begin{aligned}&\leq \|\partial_{x_t} \ln p_\sigma(\cdot|x_t) - \partial_{x_t} \ln p(\cdot|x_t)\|_\infty^2 \\ &\quad + \mathbb{E}_{y_t\sim p(y_t|y)}[(\partial_{x_t} \ln p(y_t|x_t) - \partial_{x_t} \ln p(y|x_t))^2]\end{aligned}$$

We will bound each of the summands separately. Firstly, by Assumption B.1.9 $(y_t, x_t) \to p(y_t|x_t)$ is C^2 and therefore $(y_t, x_t) \to \partial_{x_t} p(y_t|x_t)$ is C^1. Moreover, since $\mathscr{X}$ is compact, $y_t \to \partial_{x_t} p(y_t|x_t)$ is Lipschitz for some Lipschitz constant L. Therefore, by Lemma B.1.14,

$$\mathbb{E}_{y_t\sim p(y_t|y)}[(\partial_{x_t} \ln p(y_t|x_t) - \partial_{x_t} \ln p(y|x_t))^2] \leq L^2 \sigma^y(t)^2.$$

To finish the proof, we need to bound

$$\|\partial_{x_t} \ln p_\sigma(\cdot|x_t) - \partial_{x_t} \ln p(\cdot|x_t)\|_\infty^2$$

First, we apply the chain rule

$$\|\partial_{x_t} \ln p_\sigma(\cdot|x_t) - \partial_{x_t} \ln p(\cdot|x_t)\|_\infty^2 = \left\| \frac{\partial_{x_t} p_\sigma(\cdot|x_t)}{p_\sigma(\cdot|x_t)} - \frac{\partial_{x_t} p(\cdot|x_t)}{p(\cdot|x_t)} \right\|_\infty^2.$$

Adding and subtracting $\frac{\partial_{x_t} p_\sigma(\cdot|x_t)}{p(\cdot|x_t)}$:

$$\leq \left\| \frac{\partial_{x_t} p_\sigma(\cdot|x_t)}{p_\sigma(\cdot|x_t)} - \frac{\partial_{x_t} p_\sigma(\cdot|x_t)}{p(\cdot|x_t)} \right\|_\infty^2 + \left\| \frac{\partial_{x_t} p_\sigma(\cdot|x_t)}{p(\cdot|x_t)} - \frac{\partial_{x_t} p(\cdot|x_t)}{p(\cdot|x_t)} \right\|_\infty^2$$
$$= \left\| \frac{\partial_{x_t} p_\sigma(\cdot|x_t)[p(\cdot|x_t) - p_\sigma(\cdot|x_t)]}{p_\sigma(\cdot|x_t) p(\cdot|x_t)} \right\|_\infty^2 + \left\| \frac{\partial_{x_t} p_\sigma(\cdot|x_t) - \partial_{x_t} p(\cdot|x_t)}{p(\cdot|x_t)} \right\|_\infty^2$$

By assumption B.1.9 and B.1.11 we have that $\partial_{x_t} p_\sigma(\cdot|x_t)$, $p_\sigma(\cdot|x_t)$ and $p(\cdot|x_t)$ are continuous functions on a compact domain. Therefore, $\partial_{x_t} p_\sigma(\cdot|x_t)$ is bounded from above by some constant M. Moreover, by adding assumption B.1.10 we obtain that $p_\sigma(\cdot|x_t)$ and $p(\cdot|x_t)$ are bounded from below by some $\varepsilon > 0$. Therefore

$$\leq \left\| \frac{\partial_{x_t} p_\sigma(\cdot|x_t)[p(\cdot|x_t) - p_\sigma(\cdot|x_t)]}{p_\sigma(\cdot|x_t) p(\cdot|x_t)} \right\|_\infty^2 + \left\| \frac{\partial_{x_t} p_\sigma(\cdot|x_t) - \partial_{x_t} p(\cdot|x_t)}{p(\cdot|x_t)} \right\|_\infty^2$$
$$\leq \frac{M}{\varepsilon^2} \left\| p(\cdot|x_t) - p_\sigma(\cdot|x_t) \right\|_\infty^2 + \frac{1}{\varepsilon} \left\| \partial_{x_t} p_\sigma(\cdot|x_t) - \partial_{x_t} p(\cdot|x_t) \right\|_\infty^2$$

Now by Lemma B.1.13 and B.1.12

$$\leq \frac{M}{\varepsilon^2} E_1(1/\sigma^y(t)^2) + \frac{1}{\varepsilon} E_2(1/\sigma^y(t)^2)$$

where E_1 and E_2 are monotonically decreasing to zero. The theorem follows with $E(1/\sigma^y(t)^2) := \frac{M}{\varepsilon^2} E_1(1/\sigma^y(t)^2) + \frac{1}{\varepsilon} E_2(1/\sigma^y(t)^2) + L^2 \sigma^y(t)^2$, which monotonically decreases to zero as $\sigma^y(t)^2$ decreases to zero. □

B.2 Architectures and hyperparameters

We used almost the same neural network architecture across all tasks and all estimators, so that we can compare the estimators fairly. The only difference between the score model for the diffusive estimators and the score model for the CDE estimator is that the former contains 6 instead 3 filters in the final convolution to account for the joint score estimation. This difference in the final convolution leads to negligible difference in the number of parameters, which is highly unlikely to have impacted the final performance.

We used the basic version of the DDPM architecture with the following hyperparameters: channel dimension 96, depth multipliers $[1,1,2,2,3,3]$, 2 ResNet Blocks per scale and attention in the final 3 scales. The total parameter count is 43.5M. Song et al. [186] report improved performance with the NCSN++ architecture over the baseline DDPM when training with the VE SDE. This claim is also supported by the work of Saharia et al. [167]. Therefore,

adopting this architecture is likely to improve the performance of all estimators and lead to even more competitive performance over state-of-the-art methods. For all estimators, we concatenate the condition image y or $y(t)$ with the diffused target $x(t)$ and pass the concatenated image as input to the score model for score calculation. In the super-resolution experiment, we first interpolate the condition to the same resolution as the target using nearest neighbours interpolation and then concatenate it with the target image.

We used exponential moving average (EMA) with rate 0.999 and the same optimizer settings as in [186]. Moreover, we used a batch size of 50 for the super-resolution and edge to image translation experiments and a batch size of 100 for the inpainting experiments.

B.3 Extended visual results

We provide additional samples in Figures B.1, B.2, B.3, B.4, B.5 and B.7.

B.4 Potential negative impact

The potential of negative impact of this work is the same as that of any work that advances generative modeling. Generative modeling can be used for the creation of deep-fakes which can be used for malicious purposes such as disinformation and blackmailing. However, research on generative modeling can indirectly or directly contribute to the robustification of deep-fake detection algorithms. Moreover, generative models have proven very useful in academic research and in industry. The potential benefits of generative modeling outweigh the potential threats. Therefore, the research community should continue to conduct research on generative modeling.

Fig. B.1 Vanilla - CelebA-HQ 128×128

Fig. B.2 Multiscale - CelebA-HQ 128×128

Fig. B.3 Multiscale + - CelebA-HQ 128 × 128

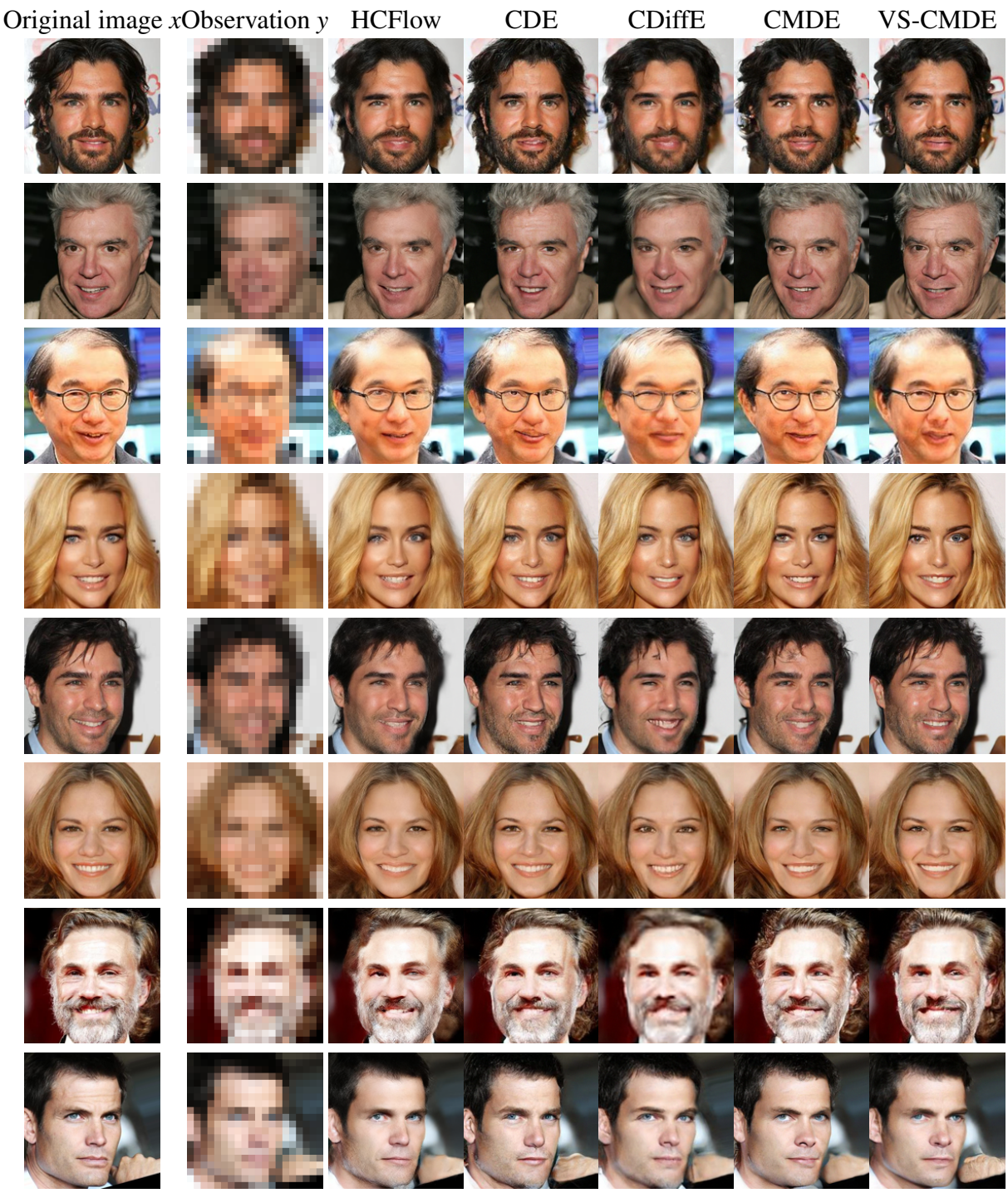


Fig. B.4 Extended super-resolution results.

Fig. B.5 Extended inpainting results.

Fig. B.6 Extended edge to shoe synthesis results.

Fig. B.7 Extended edge to shoe synthesis results.

Appendix C

Diffusion Models Encode the Intrinsic Dimension of Data Manifolds

C.1 Extended background on diffusion models

Setup: In [183] score-based [85] and diffusion-based [178, 78] generative models have been unified into a single continuous-time score-based framework where the diffusion is driven by a stochastic differential equation. This framework relies on Anderson's Theorem [2], which states that under certain Lipschitz conditions on the drift coefficient $f : \mathbb{R}^{n_x} \times \mathbb{R} \to \mathbb{R}^{n_x}$ and on the diffusion coefficient $G : \mathbb{R}^{n_x} \times \mathbb{R} \to \mathbb{R}^{n_x} \times \mathbb{R}^{n_x}$ and an integrability condition on the target distribution $p_0(\mathbf{x}_0)$ a forward diffusion process governed by the following SDE:

$$d\mathbf{x}_t = f(\mathbf{x}_t, t)dt + G(\mathbf{x}_t, t)d\mathbf{w}_t \tag{C.1}$$

has a reverse diffusion process governed by the following SDE:

$$d\mathbf{x}_t = [f(\mathbf{x}_t, t) - G(\mathbf{x}_t, t)G(\mathbf{x}_t, t)^T \nabla_{\mathbf{x}_t} \ln p_t(\mathbf{x}_t)]dt + G(\mathbf{x}_t, t)d\bar{\mathbf{w}}_t, \tag{C.2}$$

where $\bar{\mathbf{w}}_t$ is a standard Wiener process in reverse time.

The forward diffusion process transforms the *target distribution* $p_0(\mathbf{x}_0)$ to a *diffused distribution* $p_T(\mathbf{x}_T)$ after diffusion time T. By appropriately selecting the drift and the diffusion coefficients of the forward SDE, we can make sure that after sufficiently long time T, the diffused distribution $p_T(\mathbf{x}_T)$ approximates a simple distribution, such as $\mathcal{N}(\mathbf{0}, \mathbf{I})$. We refer to this simple distribution as the *prior distribution*, denoted by π. The reverse diffusion process transforms the diffused distribution $p_T(\mathbf{x}_T)$ to the data distribution $p_0(\mathbf{x}_0)$ and the prior distribution π to a distribution p^{SDE}. The distribution p^{SDE} is close to $p_0(\mathbf{x}_0)$ if the

diffused distribution $p_T(\mathbf{x}_T)$ is close to the prior distribution π. We get samples from p^{SDE} by sampling from π and simulating the reverse SDE from time T to time 0.

Sampling: To get samples by simulating the reverse SDE, we need access to the time-dependent *score function* $\nabla_{\mathbf{x}_t} \ln p_t(\mathbf{x}_t)$. In practice, we approximate the time-dependent score function with a neural network $s_\theta(\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}_t} \ln p_t(\mathbf{x}_t)$ and simulate the reverse SDE presented in equation C.3 to map the prior distribution π to p_θ^{SDE}.

$$d\mathbf{x}_t = [f(\mathbf{x}_t, t) - G(\mathbf{x}_t, t)G(\mathbf{x}_t, t)^T s_\theta(\mathbf{x}_t, t)]dt + G(\mathbf{x}_t, t)d\bar{\mathbf{w}}_t, \tag{C.3}$$

If the prior distribution is close to the diffused distribution and the approximated score function is close to the ground truth score function, the modeled distribution p_θ^{SDE} is provably close to the target distribution $p_0(\mathbf{x}_0)$. This statement is formalised in the language of distributional distances in the work of [181].

Training: A neural network $s_\theta(\mathbf{x}_t, t)$ can be trained to approximate the score function $\nabla_{\mathbf{x}_t} \ln p_t(\mathbf{x}_t)$ by minimizing the weighted score matching objective

$$\mathscr{L}_{SM}(\theta, \lambda(\cdot)) := \frac{1}{2} \mathbb{E}_{\substack{t \sim U(0,T) \\ \mathbf{x}_t \sim p_t(\mathbf{x}_t)}} [\lambda(t) \| \nabla_{\mathbf{x}_t} \ln p_t(\mathbf{x}_t) - s_\theta(\mathbf{x}_t, t) \|_2^2] \tag{C.4}$$

where $\lambda : [0, T] \to \mathbb{R}_+$ is a positive weighting function.

However, the above quantity cannot be optimized directly since we don't have access to the ground truth score $\nabla_{\mathbf{x}_t} \ln p_t(\mathbf{x}_t)$. Therefore in practice, a different objective has to be used [85, 206, 183]. In [183], the weighted denoising score-matching objective is used, which is defined as

$$\mathscr{L}_{DSM}(\theta, \lambda(\cdot)) := \frac{1}{2} \mathbb{E}_{\substack{t \sim U(0,T) \\ \mathbf{x}_0 \sim p_0(\mathbf{x}_0) \\ \mathbf{x}_t \sim p_t(\mathbf{x}_t | \mathbf{x}_0)}} [\lambda(t) \| \nabla_{\mathbf{x}_t} \ln p_t(\mathbf{x}_t | \mathbf{x}_0) - s_\theta(\mathbf{x}_t, t) \|_2^2] \tag{C.5}$$

The difference between DSM and SM is the replacement of the ground truth score which we do not know by the score of the perturbation kernel which we know analytically for many choices of forward SDEs. The choice of the weighted DSM objective is justified because the weighted DSM objective is equal to the SM objective up to a constant that does not depend on the parameters of the model θ. The reader can refer to [206] for the proof.

C.2 Training details

We trained the score model using the weighted denoising score matching objective [183], presented in eq. C.5. We used the likelihood weighting function, i.e. $\lambda(t) = g(t)^2$, where $g(t)$ is the diffusion coefficient of the forward SDE.

C.2.1 Euclidean data

For all of our experiments on Euclidean data, we used a fully connected network with 5 hidden layers and 2048 nodes in each hidden layer to approximate the score function. The input and output dimension is the same as the ambient dimension. For the optimisation of the model, we used the Adam algorithm with a learning rate of $2e{-}5$ and exponential moving average (EMA) on the weights of the model with a decay rate of 0.9999. Moreover, we chose the variance exploding SDE [183] as the forward process with $\sigma_{min} = 0.01$ and $\sigma_{max} = 4$.

C.2.2 Image data

For all our of our experiments on image data, we used the DDPM architecture [77] with variance exploding SDE [183] and hyperparameters indicated in Table C.1.

C.2.3 Auto-encoder

The encoder and decoder encoder architectures are based on the DDPM U-Net [78], which we call half-U nets.

For the encoder we used the downsampling part of the U-Net and removed the upsampling part and the skip connections. The downscaled tensor is flattened and mapped to the latent dimension with an additional linear layer.

For the decoder we start by linearly transforming the latent vector and reshaping it into a tensor of appropriate dimension. Then we used the upsampling part of the DDPM U-Net.

We used the Adam optimizer and EMA rate 0.999. We used learning rate scheduler reducing the loss on plateau starting form 10^{-4} and stopping at 10^{-5}. We trained the auto-encoder for each latent dimension for 36h on NVIDIA A-100 GPU. At the end we used checkpoints which minimized the validation loss to evaluate the reconstruction error.

All other hyperparameters are included in Table C.1.

Hyper-parameter	**MNIST**	**Synthetic Image data**
Number of filters	128	128
Channel multipliers	(1, 2, 2, 4)	(1, 2, 2, 2)
Dropout	0.1	0.1
EMA rate	0.999	0.999
Normalization	GroupNorm	GroupNorm
Nonlinearity	Swish	Swish
Number of residual blocks	4	4
Attention resolution	16	16
Convolution size	3	3
σ_{min}	0.009	0.01
σ_{max}	50	50
Learning rate	Scheduler(10^{-4}, 10^{-5})	$2 \cdot 10^{-4}$

Table C.1 DDPM Model Parameters

C.3 Benchmarking

We compared our method against well established approaches to intrinsic dimensionality estimation: the MLE estimator [117], [69], Local PCA [52] and Probabilistic PCA [131] [19]. For MLE estimator and local PCA we used the implementation provided in the R package INTRINSICDIMENSION [90]. The MLE estimator has an important hyperparameter m - the number of nearest neighbour distances that should be used for the dimension estimation. We used values $m = 5$ and 20 since these are extremal values considered in [154]. For PPCA we used the SCIKIT-LEARN implementation [148]. The code for reproducing the benchmarking experiments is included in our codebase.

For the ID-NF method [80], we used the official implementation available at https://github.com/chrvt/ID-NF. For the Euclidean data, we utilized the "vector data" folder, which employs block neural autoregressive flows to learn the normalizing flows, from which the intrinsic dimension is extracted using the ID-NF method. For the synthetic image data and MNIST, we used the "images" folder, which uses rational quadratic splines to train the normalizing flows.

C.4 Proofs

Here we provide full proofs for the statements in Section 5.4. First, we show that for any point $\mathbf{x}$ sufficiently close to the data manifold and sufficiently small t the score $\nabla_{\mathbf{x}} \ln p_t(\mathbf{x})$ points directly at the manifold. We demonstrate this by showing that projection of the score

in any direction $v \perp \mathbf{n}$ vanishes in proportion to the projection on $\mathbf{n} = \frac{(\pi(\mathbf{x})-\mathbf{x})}{\|\pi(\mathbf{x})-\mathbf{x}\|}$ as $t \to 0$. Then Theorem 5.4.1 and Corollary 5.4.2 will follow easily from this result.

Theorem C.4.1. *Suppose that the the support of the data distribution P_0 is contained in a compact embedded sub-manifold $\mathscr{M} \subseteq \mathbb{R}^d$ and let P_t be the distribution of samples from P_0 diffused for time t. Then, under mild assumptions, for any point $\mathbf{x} \in \mathbb{R}^d$ sufficiently close to $\mathscr{M}$, with orthogonal projection on $\mathscr{M}$, given by $\pi(\mathbf{x})$. Let $\mathbf{n}$ be a unit vector pointing from $\mathbf{x}$ to $\pi(\mathbf{x})$, then we have that for any unit vector v orthogonal to $\mathbf{n}$:*

$$\frac{v^T \nabla_{\mathbf{x}} \ln p_t(\mathbf{x})}{\mathbf{n}^T \nabla_{\mathbf{x}} \ln p_t(\mathbf{x})} \to 0, as\ t \to 0.$$

Assumptions

1. The distribution P_0 has a smooth density p_0 wrt the volume measure on the manifold.
2. The density p_0 is bounded away from zero on the manifold.

Illustrative simple case

We first present an illustrative proof of a simple case of $\mathscr{M}$ being a linear subspace with $k = 1$ and $d = 2$. This case gives all of the essential ideas behind the general proof without much of the technicality. For the more interested reader, we then provide a proof of the result for a general manifold, using tools such as the notion of tubular neighbourhoods and some results from Morse theory.

Without the loss of generality assume that $\mathscr{M} = \{(x_1, x_2) \in \mathbb{R}^2 : x_2 = 0\}$ the line given by the x_1-axis. Pick a point $\mathbf{x} \in \mathbb{R}^2$. The score at point $\mathbf{x}$ is given by

$$\nabla_{\mathbf{x}} \ln p_t(\mathbf{x}) = \frac{1}{\sigma_t^2 p_t(\mathbf{x})} \int_{\mathscr{M}} (\mathbf{y} - \mathbf{x}) \mathscr{N}(\mathbf{y}|\mathbf{x}, \sigma_t^2 \mathbf{I}) p_0(\mathbf{y}) d\mathbf{y}.$$

Notice that $\mathscr{N}((y_1, y_1)|(x_1, x_2), \sigma_t^2 \mathbf{I})$[1] is a bivariate normal distribution and its restriction to $\mathscr{M}$ is equal to $\mathscr{N}((y_1, 0)|(x_1, x_2), \sigma_t^2 \mathbf{I}) = \mathscr{N}(0|x_2, \sigma_t^2) \mathscr{N}(y_1|x_1, \sigma_t^2)$. Therefore

$$\nabla_{\mathbf{x}} \ln p_t(\mathbf{x}) = \frac{\mathscr{N}(0|x_2, \sigma_t^2)}{\sigma_t^2 p_t(\mathbf{x})} \int_{\mathscr{M}} (\mathbf{y} - \mathbf{x}) \mathscr{N}(y_1|x_1, \sigma_t^2) p_0(\mathbf{y}) d\mathbf{y}.$$

This means that the score is the weighted average of vectors pointing from $\mathbf{x}$ to $\mathbf{y}$ over all choices of points $\mathbf{y}$ on the manifold, with weights given by $w(\mathbf{y}; \sigma_t) := \mathscr{N}(y_1|x_1, \sigma_t^2) p_0(\mathbf{y})$

[1] In component-wise notation $\mathbf{y} = (y_1, y_2)$ and $\mathbf{x} = (x_1, x_2)$.

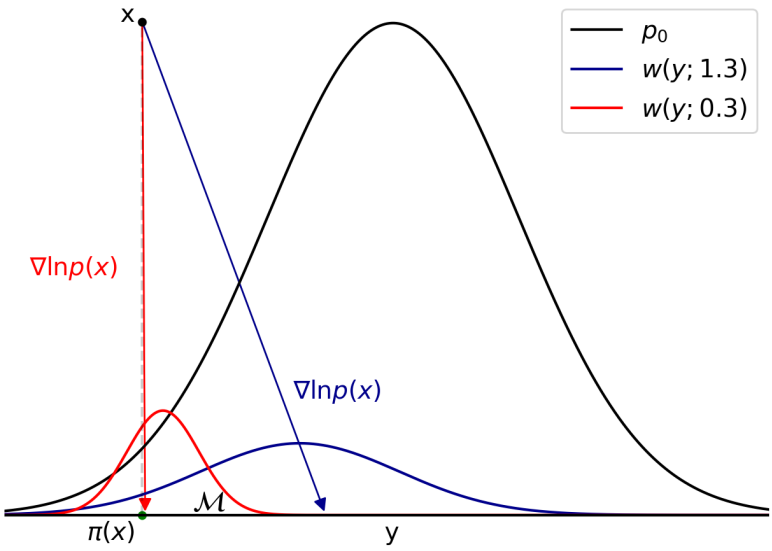


Fig. C.1 The score is the weighted average of vectors pointing from $\mathbf{x}$ to $\mathbf{y}$ with weights given by $w(\mathbf{y};\sigma_t)$. As σ_t decreases weights $w(\mathbf{y};\sigma_t)$ concentrate around $\pi(\mathbf{x})$ and the influence of points $\mathbf{y}$ away from projection $\pi(\mathbf{x})$ becomes insignificant. Therefore, the direction of the score tends to a align with $\pi(\mathbf{x})-\mathbf{x}$. (Norm of the score vectors on the figure was scaled for better visibility. The direction is preserved.)

(see Figure C.1 for visual explanation). For small σ_t these weights concentrate around $\pi(\mathbf{x}) = (x_1, 0)$ the projection of $\mathbf{x}$ on $\mathscr{M}$, and vanishing far away from it. Consider a ratio of the tangential part to the normal part of the score:

$$\frac{v^T \nabla_{\mathbf{x}} \ln p_t(\mathbf{x})}{\mathbf{n}^T \nabla_{\mathbf{x}} \ln p_t(\mathbf{x})} = \frac{\int_{\mathscr{M}} v^T(\mathbf{y}-\mathbf{x}) \mathscr{N}(y_1|x_1,\sigma_t^2) p_0(\mathbf{y}) d\mathbf{y}}{\int_{\mathscr{M}} \mathbf{n}^T(\mathbf{y}-\mathbf{x}) \mathscr{N}(y_1|x_1,\sigma_t^2) p_0(\mathbf{y}) d\mathbf{y}} \xrightarrow[\sigma_t \to 0]{} \frac{\int_{\mathscr{M}} v^T(\mathbf{y}-\mathbf{x}) \delta_{x_1}(y_1) p_0(\mathbf{y}) d\mathbf{y}}{\int_{\mathscr{M}} \mathbf{n}^T(\mathbf{y}-\mathbf{x}) \delta_{x_1}(y_1) p_0(\mathbf{y}) d\mathbf{y}}$$
$$= \frac{v^T((x_1,0)-\mathbf{x})}{\mathbf{n}^T((x_1,0)-\mathbf{x})} = \frac{(1,0)^T(0,-x_2)}{(0,1)^T(0,-x_2)} = 0$$

where v and $\mathbf{n}$ are unit vectors in tangential and normal directions respectively. This implies,

$$\mathrm{S}_{\cos}(\mathbf{n}, \nabla_{\mathbf{x}} \ln p_t(\mathbf{x})) = \frac{\mathbf{n}^T \nabla_{\mathbf{x}} \ln p_t(\mathbf{x})}{\|\nabla_{\mathbf{x}} \ln p_t(\mathbf{x})\|} = \frac{\mathbf{n}^T \nabla_{\mathbf{x}} \ln p_t(\mathbf{x})}{\sqrt{(\mathbf{n}^T \nabla_{\mathbf{x}} \ln p_t(\mathbf{x}))^2 + (v^T \nabla_{\mathbf{x}} \ln p_t(\mathbf{x}))^2}}$$
$$= \frac{1}{\sqrt{1 + \left(\frac{v^T \nabla_{\mathbf{x}} \ln p_t(\mathbf{x})}{\mathbf{n}^T \nabla_{\mathbf{x}} \ln p_t(\mathbf{x})}\right)^2}} \xrightarrow[t \to 0]{} 1.$$

This establishes the theorem for the simple case. The corollary follows immediately since in the simple case we have $\mathbf{T} = v^T$ and $\mathbf{N} = \mathbf{n}^T$. □

Deriving the formula for the density of P_t

Let $\mathscr{M}$ be a compact k-dimensional manifold embedded in $\mathbb{R}^d$. Let $A \subseteq \mathbb{R}^d$. We define the measure P_0 on $\mathbb{R}^d$ as

$$P_0(A) := \int_{A \cap M} p_0(\mathbf{y}) d\mathbf{y} \tag{C.6}$$

where $d\mathbf{y}$ is the volume form[2] on M and p_0 is a smooth function on $\mathscr{M}$ [3] such that $\int_{\mathscr{M}} p_0(\mathbf{y}) d\mathbf{y} = 1$. Let $f : \mathbb{R}^d \to \mathbb{R}$ be a P_0-measurable function. By approximating f with simple functions (linear combination of indicator functions) we conclude that:

$$\int_A f dP_0 = \int_{A \cap M} f(\mathbf{y}) p_0(\mathbf{y}) d\mathbf{y} \tag{C.7}$$

Consider a measure P_t as a convolution of P_0 with a normal distribution on $\mathbb{R}^d$. For any measurable $A \subseteq \mathbb{R}^d$ we have

$$\begin{aligned}
(P_0 * \mathscr{N}_{0,t})(A) &:= \int_{\mathbb{R}^d} \int_{A-\mathbf{y}} d\mathscr{N}_{0,t}(\mathbf{x}) dP_0(\mathbf{y}) = \int_{\mathbb{R}^d} \int_{A-\mathbf{y}} \mathscr{N}(\mathbf{x}|0, \sigma_t^2 \mathbf{I}) d\mathbf{x} dP_0(\mathbf{y}) \\
&= \int_{\mathbb{R}^d} \int_A \mathscr{N}(\mathbf{x} - \mathbf{y}|0, \sigma_t^2 \mathbf{I}) d\mathbf{x} dP_0(\mathbf{y}) = \int_{\mathbb{R}^d} \int_A \mathscr{N}(\mathbf{y}|\mathbf{x}, \sigma_t^2 \mathbf{I}) d\mathbf{x} dP_0(\mathbf{y}) \\
&= \int_A \int_{\mathbb{R}^d} \mathscr{N}(\mathbf{y}|\mathbf{x}, \sigma_t^2 \mathbf{I}) dP_0(\mathbf{y}) d\mathbf{x} \overset{\text{(C.7)}}{=} \int_A \int_{\mathscr{M}} \mathscr{N}(\mathbf{y}|\mathbf{x}, \sigma_t^2 \mathbf{I}) p_0(\mathbf{y}) d\mathbf{y} d\mathbf{x}
\end{aligned}$$

where $d\mathbf{y}$ is a volume form on M and $d\mathbf{x}$ is a volume form on $\mathbb{R}^d$. Therefore the measure P_t has a density on $\mathbb{R}^d$ given by:

$$p_t(\mathbf{x}) = \int_{\mathscr{M}} \mathscr{N}(\mathbf{y}|\mathbf{x}, \sigma_t^2 \mathbf{I}) p_0(\mathbf{y}) d\mathbf{y}. \tag{C.8}$$

Note the $\mathscr{N}(\mathbf{y}|\mathbf{x}, \sigma_t^2 \mathbf{I})$ here. Typically one would write this as $\mathscr{N}(\mathbf{x}|\mathbf{y}, \sigma_t^2 \mathbf{I})$ and think of (C.8) as the quantity of probability mass at point $\mathbf{x}$ after diffusing for time t with initial distribution $p_0(\mathbf{y})$. We instead write it this way as it will be more intuitive to think of (C.8) as the average probability mass that intersects the manifold after diffusing from a delta distribution at $\mathbf{x}$ (where the average is taken over $p_0(\mathbf{y})$). These are of course equivalent as $\mathscr{N}(\mathbf{x}|\mathbf{y}, \sigma_t^2 \mathbf{I})$ is symmetric is $\mathbf{x}$ and $\mathbf{y}$.

[2] Integrating over A using the volume form of $\mathscr{M}$ can be thought of as taking an appropriately re-scaled Lebesgue integral over A. That is $P_0(A) = \int_{A \cap \mathscr{M}} p_0(\mathbf{y}) d\mathbf{y} = \int_A \int_{\mathbb{R}^d} \delta(\mathbf{s} - \mathbf{y}) \hat{p}_0(\mathbf{s}) d\mathbf{s} d\mathbf{y}$. Where the latter are Lebesgue integrals and $\hat{p}_0(\mathbf{s}) = p_0(\mathbf{s})$ for $\mathbf{s} \in \mathscr{M}$ and zero otherwise.

[3] i.e. for any chart $\phi : \mathbb{R}^k \supseteq U \to M$ the composition $p_0 \circ \phi : \mathbb{R}^k \supseteq U \to \mathbb{R}$ is smooth.

Tubular Neighbourhoods

First we need to ensure that the point $\mathbf{x}$ has a unique projection on $\mathscr{M}$. This is always true for an $\mathbf{x}$ sufficiently close to $\mathscr{M}$. We can formalize this with the notion of *tubular neighbourhood* - a tube around $\mathscr{M}$ such that every point $\mathbf{x}$ inside can be uniquely represented as a sum of the point on the manifold and a vector from the normal bundle i.e. $\mathbf{x} = \mathbf{y} + \mathbf{v}$ where $\mathbf{y} \in \mathscr{M}$ and $\mathbf{v} \in \mathscr{N}_{\mathbf{y}}\mathscr{M}$. Formally:

Definition C.4.2. Endpoint Map
The endpoint map $Y : \mathscr{N}\mathscr{M} \to \mathbb{R}^d$ is defined by $Y(\mathbf{y}, \mathbf{v}) = \mathbf{y} + \mathbf{v}$ for $\mathbf{y} \in \mathscr{M}$ and $\mathbf{v} \in \mathscr{N}_{\mathbf{y}}\mathscr{M}$.

Definition C.4.3. Tubular Neighbourhood
A (uniform) tubular neighbourhood of $\mathscr{M}$ is a neighbourhood U_R of $\mathscr{M}$ in $\mathbb{R}^d$ that is a diffeomorphic image under the endpoint map of an open subset $V_R \subseteq \mathscr{N}\mathscr{M}$ of the form:

$$V_R = \{(\mathbf{y}, \mathbf{v}) \in \mathscr{N}\mathscr{M} : \|\mathbf{v}\|_2 < R\}$$

Since Y restricted to V_R is a diffeomorphism[4], it follows that every point $\mathbf{x} = (\mathbf{y}, \mathbf{v})$ in the tubular neighbourhood has a unique orthogonal projection on $\mathscr{M}$ given by $\mathbf{y}$. We will denote this projection as $\pi(\mathbf{x})$.

Conveniently, it turns out that every compact embedded submanifold of $\mathbb{R}^d$ has a tubular neighborhood.

Theorem C.4.4. *Tubular Neighborhood Theorem [Theorem 5.25][115]*
Every compact embedded submanifold of $\mathbb{R}^d$ has a uniform tubular neighborhood.

Preliminary lemmas and Morse theory

In this section we will establish that for every $\mathbf{x}$ in the tubular neighbourhood of $\mathscr{M}$ there exists an open neighbourhood E of $\pi(\mathbf{x})$ such that:

1. $\mathbf{n}^T(\mathbf{y} - \mathbf{x}) > \|\pi(\mathbf{x}) - \mathbf{x}\|_2 - \varepsilon$ on E .
2. $|v^T(\mathbf{y} - \mathbf{x})| < \varepsilon$ on E.
3. The mass of a Gaussian centred at $\mathbf{x}$ is concentrated in E,

$$\frac{\int_{\mathscr{M}\setminus E} \mathscr{N}(\mathbf{y}|\mathbf{x}, \sigma_t^2\mathbf{I})d\mathbf{y}}{\int_E \mathscr{N}(\mathbf{y}|\mathbf{x}, \sigma_t^2\mathbf{I})d\mathbf{y}} \to 0 \text{ as } t \to 0.$$

[4] so in particular it is a bijection

We begin by defining an E which satisfies the first two conditions.

Lemma C.4.5. *Choose E contained in a ball of radius $0 < \varepsilon < \|\mathbf{x} - \pi(\mathbf{x})\|$ around $\pi(\mathbf{x})$. Let $\mathbf{y} \in E$, and let $\mathbf{v}_\varepsilon := \mathbf{y} - \pi(\mathbf{x})$. Then*

1. $\mathbf{n}^T(\mathbf{y}-\mathbf{x}) > \|\pi(\mathbf{x})-\mathbf{x}\|_2 - \varepsilon$ *on E .*
2. $|\nu^T(\mathbf{y}-\mathbf{x})| < \varepsilon$ *on E.*

Proof. By direct computation

$$\begin{aligned}\mathbf{n}^T(\mathbf{y}-\mathbf{x}) &= \mathbf{n}^T((\mathbf{x}-\pi(\mathbf{x}))+\mathbf{v}_\varepsilon)\\ &= \mathbf{n}^T(\mathbf{x}-\pi(\mathbf{x}))+\mathbf{n}^T\mathbf{v}_\varepsilon\\ &= \|\mathbf{x}-\pi(\mathbf{x})\|_2+\mathbf{n}^T\mathbf{v}_\varepsilon\\ &\geq \|\mathbf{x}-\pi(\mathbf{x})\|-\|\mathbf{v}_\varepsilon\|_2.\end{aligned}$$

We have that $\|\mathbf{v}_\varepsilon\| < \varepsilon$, hence for all $\mathbf{y}$ in E we have that $\mathbf{n}^T(\mathbf{y}-\mathbf{x}) \geq \|\mathbf{x}-\pi(\mathbf{x})\| - \varepsilon > 0$. For the second inequality

$$\begin{aligned}|\nu^T(\mathbf{y}-\mathbf{x})| &\leq |\nu^T((\mathbf{x}-\pi(\mathbf{x}))|+|\nu^T\mathbf{v}_\varepsilon|\\ &\leq \|\mathbf{v}_\varepsilon\|_2\\ &\leq \varepsilon.\end{aligned}$$

□

Now to find E which also satisfies the last condition we proceed by recalling some elementary definitions and results of Morse theory.

Theorem C.4.6. *Morse lemma [Corollary 1.17][136]*
If $\mathbf{y}_0$ is a non-degenerate critical point of index γ of a smooth function $f : \mathscr{M} \to \mathbb{R}$, then there exist a chart $\phi = (\phi_i)_{i=1}^k$ in a neighbourhood U of $\mathbf{y}_0$ such that $\phi(\mathbf{y}_0) = 0$, and in this chart we have the equality:

$$f(\mathbf{y}) = f(\mathbf{y}_0) - \sum_{i=1}^{\gamma}\phi_i(\mathbf{y})^2 + \sum_{i=\gamma+1}^{k}\phi_i(\mathbf{y})^2$$

Let $f_\mathbf{x}(\mathbf{y}) : \mathscr{M} \to \mathbb{R}$ denote the squared distance function from $\mathbf{x}$ given by $f_\mathbf{x}(\mathbf{y}) = \|\mathbf{x}-\mathbf{y}\|_2^2$. We will establish that if $\mathbf{x}$ is in a tubular neighbourhood, then its projection $\pi(\mathbf{x})$ is a non-degenerate critical point of $f_\mathbf{x}$ of index zero.

Definition C.4.7. Focal point
A point $\mathbf{x} = Y(\mathbf{y}, \mathbf{v})$ in the image of the endpoint map Y, is called a non-focal point of $\mathscr{M}$ with respect to $\mathbf{y}$ if $dY(\mathbf{y}, \mathbf{v})$ is an isomorphism. Otherwise it is called a focal point.

Theorem C.4.8. *Critical points and focal points [145]*
Let $\mathscr{M}$ be an embedded submanifold of $\mathbb{R}^d$, $\mathbf{y} \in M$, $\mathbf{v} \in \mathscr{N}_{\mathbf{y}}\mathscr{M}$, and $\mathbf{x} = Y(\mathbf{y}, \mathbf{v}) = \mathbf{y} + \mathbf{v}$. Then

1. *$\mathbf{y}$ is a critical point of $f_{\mathbf{x}}$.*
2. *$\mathbf{y}$ is a non-degenerate critical point of $f_{\mathbf{x}}$ if and only if $\mathbf{x}$ is a non-focal point.*
3. *Index of f_a at $\mathbf{y}$ is equal to the number of focal points of $\mathscr{M}$ with respect to $\mathbf{y}$ on the line segment joining $\mathbf{y}$ to $\mathbf{x}$.*

Because the restriction of the endpoint map Y to the tubular neighbourhood is a diffeomorphism, the differential dY is an isomorphism for every point in the tubular neighbourhood. Therefore there are no focal points of $\mathscr{M}$ in the tubular neighbourhood. Hence, it follows directly from the above theorem, that if $\mathbf{x}$ is in the tubular neighbourhood, then the projection $\pi(\mathbf{x})$ is a non-degenerate critical point of $f_{\mathbf{x}}$ of index zero. Now we are ready to prove the following lemma.

Lemma C.4.9. *There exists a connected open neighbourhood E of $\pi(\mathbf{x})$ satisfying conditions of lemma C.4.5 and such that,*

$$\frac{\int_{\mathscr{M}\setminus E} \mathscr{N}(\mathbf{y}|\mathbf{x}, \sigma_t^2\mathbf{I})d\mathbf{y}}{\int_E \mathscr{N}(\mathbf{y}|\mathbf{x}, \sigma_t^2\mathbf{I})d\mathbf{y}} \to 0 \text{ as } t \to 0. \tag{C.9}$$

Proof. Fix $\varepsilon > 0$. Then conditions of lemma C.4.5 are satisfied inside $B(\pi(\mathbf{x}), \varepsilon)$. We have demonstrated that $\pi(\mathbf{x})$ fulfills the criteria stipulated by the Morse lemma. Consequently, we can pick $\tilde{U}$ as the neighborhood and ϕ as the coordinate system that the Morse lemma provides. Now let $U = \tilde{U} \cap B(\pi(\mathbf{x}), \varepsilon)$. Since $\mathscr{M}$ is compact and U is open, there exists $m = \min_{\mathscr{M}\setminus U} f_{\mathbf{x}}(\mathbf{y})$ and by uniqueness of projection $f_{\mathbf{x}}(\pi(\mathbf{x})) < m$. Let $r = \sqrt{m - (f_{\mathbf{x}}(\pi(\mathbf{x})))/2}$ and let $E = \phi^{-1}(B(0, r))$. For all $\mathbf{y} \in E$ we have:

$$f_{\mathbf{x}}(\mathbf{y}) = f_{\mathbf{x}}(\pi(\mathbf{x})) + \|\phi(\mathbf{y})\|^2 < f_{\mathbf{x}}(\pi(\mathbf{x})) + r^2 < m.$$

Notice that for every $\mathbf{y} \in U \setminus E$ we have $f_{\mathbf{x}}(\mathbf{y}) \geq f_{\mathbf{x}}(\pi(\mathbf{x})) + r^2$. Therefore we have established that

$$\forall_{\mathbf{y}\in E} \forall_{\tilde{\mathbf{y}}\in \mathscr{M}\setminus E} f_{\mathbf{x}}(\mathbf{y}) < f_{\mathbf{x}}(\tilde{\mathbf{y}}). \tag{C.10}$$

Computing directly, we have

$$\frac{\int_{\mathscr{M}\setminus E}\mathscr{N}(\mathbf{y}|\mathbf{x},\sigma_t^2\mathbf{I})d\mathbf{y}}{\int_E\mathscr{N}(\mathbf{y}|\mathbf{x},\sigma_t^2\mathbf{I})d\mathbf{y}}=\frac{\int_{\mathscr{M}\setminus E}\exp\{-f_{\mathbf{x}}(\mathbf{y})/2\sigma_t^2\}d\mathbf{y}}{\int_E\exp\{-f_{\mathbf{x}}(\mathbf{y})/2\sigma_t^2\}d\mathbf{y}} \tag{C.11}$$

By the mean value theorem there exists $\mathbf{y}^*\in E$ and $\tilde{\mathbf{y}}^*\in\mathscr{M}\setminus E$ such that

$$\int_E\exp\{-f_{\mathbf{x}}(\mathbf{y})/2\sigma_t^2\}d\mathbf{y}=\mathrm{Vol}(E)\exp\{-f_{\mathbf{x}}(\mathbf{y}^*)/2\sigma_t^2\}$$
$$\int_{\mathscr{M}\setminus E}\exp\{-f_{\mathbf{x}}(\mathbf{y})/2\sigma_t^2\}d\mathbf{y}=\mathrm{Vol}(\mathscr{M}\setminus E)\exp\{-f_{\mathbf{x}}(\tilde{\mathbf{y}}^*)/2\sigma_t^2\}$$

We can use this to evaluate (C.11) to give,

$$\begin{aligned}\frac{\int_{\mathscr{M}\setminus E}\mathscr{N}(\mathbf{y}|\mathbf{x},\sigma_t^2\mathbf{I})d\mathbf{y}}{\int_E\mathscr{N}(\mathbf{y}|\mathbf{x},\sigma_t^2\mathbf{I})d\mathbf{y}}&=\frac{\mathrm{Vol}(\mathscr{M}\setminus E)\exp\left\{-f_{\mathbf{x}}(\tilde{\mathbf{y}}^*)/2\sigma_t^2\right\}}{\mathrm{Vol}(E)\exp\left\{-f_{\mathbf{x}}(\mathbf{y}^*)/2\sigma_t^2\right\}}\\&=\frac{\mathrm{Vol}(\mathscr{M}\setminus E)}{\mathrm{Vol}(E)}\exp\left\{-\frac{f_{\mathbf{x}}(\tilde{\mathbf{y}}^*)-f_{\mathbf{x}}(\mathbf{y}^*)}{2\sigma_t^2}\right\}\end{aligned}$$

Since by (C.10) $f_{\mathbf{x}}(\tilde{\mathbf{y}}^*)-f_{\mathbf{x}}(\mathbf{y}^*)>0$ the above goes to zero as σ_t goes to zero. Moreover, since $E\subseteq U\subseteq B(\pi(\mathbf{x}),\varepsilon)$, the conditions of lemma C.4.5 are also satisfied. □

Proof of Theorem C.4.1

Fix $\varepsilon>0$. Assume that $\mathbf{x}$ is in a tubular neighbourhood of $\mathscr{M}$, so that the projection $\pi(\mathbf{x})$ exists. To simplify the expressions that appear multiple times, we define the following functions:

$$g(\mathbf{y},\mathbf{x},t)=\nu^T(\mathbf{y}-\mathbf{x})\mathscr{N}(\mathbf{y}|\mathbf{x},\sigma_t^2\mathbf{I})p_0(\mathbf{y}).$$

$$h(\mathbf{y},\mathbf{x},t)=\mathbf{n}^T(\mathbf{y}-\mathbf{x})\mathscr{N}(\mathbf{y}|\mathbf{x},\sigma_t^2\mathbf{I})p_0(\mathbf{y}),$$

Then, we have

$$\frac{\nu^T\nabla_{\mathbf{x}}\ln p_t(\mathbf{x})}{\mathbf{n}^T\nabla_{\mathbf{x}}\ln p_t(\mathbf{x})}=\frac{\nu^T\nabla_{\mathbf{x}}p_t(\mathbf{x})}{\mathbf{n}^T\nabla_{\mathbf{x}}p_t(\mathbf{x})}=\frac{\int_{\mathscr{M}}g(\mathbf{y},\mathbf{x},t)\,d\mathbf{y}}{\int_{\mathscr{M}}h(\mathbf{y},\mathbf{x},t)\,d\mathbf{y}}. \tag{C.12}$$

Split $\mathscr{M}$ into two parts: E and $\mathscr{M}\setminus E$, where $E=B(\mathbf{x},r)\cap\mathscr{M}$ is an open neighbourhood of $\pi(\mathbf{x})$ in $\mathscr{M}$ satisfying the conditions of Lemma C.4.9 and Lemma C.4.5 for a chosen $\varepsilon>0$. Then, we have that (C.12) is equal to

$$\frac{\int_E g(\mathbf{y},\mathbf{x},t)\,d\mathbf{y}}{\int_{\mathscr{M}} h(\mathbf{y},\mathbf{x},t)\,d\mathbf{y}} + \frac{\int_{\mathscr{M}\setminus E} g(\mathbf{y},\mathbf{x},t)\,d\mathbf{y}}{\int_{\mathscr{M}} h(\mathbf{y},\mathbf{x},t)\,d\mathbf{y}}. \tag{C.13}$$

We begin by bounding the first term:

$$\begin{aligned}
\frac{\int_E g(\mathbf{y},\mathbf{x},t)\,d\mathbf{y}}{\int_{\mathscr{M}} h(\mathbf{y},\mathbf{x},t)\,d\mathbf{y}} &= \frac{\int_E g(\mathbf{y},\mathbf{x},t)\,d\mathbf{y}}{\int_E h(\mathbf{y},\mathbf{x},t)\,d\mathbf{y} + \int_{\mathscr{M}\setminus E} h(\mathbf{y},\mathbf{x},t)\,d\mathbf{y}} \\
&= \frac{\dfrac{\int_E g(\mathbf{y},\mathbf{x},t)\,d\mathbf{y}}{\int_E h(\mathbf{y},\mathbf{x},t)\,d\mathbf{y}}}{1 + \dfrac{\int_{\mathscr{M}\setminus E} h(\mathbf{y},\mathbf{x},t)\,d\mathbf{y}}{\int_E h(\mathbf{y},\mathbf{x},t)\,d\mathbf{y}}} \\
&=: \frac{A_t}{1+B_t}.
\end{aligned}$$

For A_t, we have

$$|A_t| = \left|\frac{\int_E g(\mathbf{y},\mathbf{x},t)\,d\mathbf{y}}{\int_E h(\mathbf{y},\mathbf{x},t)\,d\mathbf{y}}\right|.$$

Using the fact that $\mathbf{n}^T(\mathbf{y}-\mathbf{x})$ is positive and bounded away from zero and applying the triangle inequality, we obtain

$$\begin{aligned}
|A_t| &\leq \frac{\int_E |v^T(\mathbf{y}-\mathbf{x})|\mathscr{N}(\mathbf{y}|\mathbf{x},\sigma_t^2\mathbf{I})p_0(\mathbf{y})\,d\mathbf{y}}{\int_E h(\mathbf{y},\mathbf{x},t)\,d\mathbf{y}} \\
&\leq \frac{\varepsilon\, p_{\max}\int_E \mathscr{N}(\mathbf{y}|\mathbf{x},\sigma_t^2\mathbf{I})\,d\mathbf{y}}{(\|\pi(\mathbf{x})-\mathbf{x}\|-\varepsilon)\,p_{\min}\int_E \mathscr{N}(\mathbf{y}|\mathbf{x},\sigma_t^2\mathbf{I})\,d\mathbf{y}} \\
&= \frac{p_{\max}}{p_{\min}}\frac{\varepsilon}{\|\pi(\mathbf{x})-\mathbf{x}\|-\varepsilon},
\end{aligned}$$

where in the second inequality we used that $0 < p_{\min} < p_0(\mathbf{y}) < p_{\max}$, $|v^T(\mathbf{y}-\mathbf{x})| < \varepsilon$, and $\mathbf{n}^T(\mathbf{y}-\mathbf{x}) > \|\pi(\mathbf{x})-\mathbf{x}\| - \varepsilon$ on E.

Since ε was arbitrary, this term can be made arbitrarily small.

Now we move to B_t. Let $D = \max_{\mathbf{y}\in\mathscr{M}} \|\mathbf{x}-\mathbf{y}\|$. By the triangle and Cauchy-Schwarz inequalities, we have

$$\begin{aligned}|B_t| &\leq \frac{\int_{\mathscr{M}\setminus E} |\mathbf{n}^T(\mathbf{y}-\mathbf{x})| \mathscr{N}(\mathbf{y}|\mathbf{x},\sigma_t^2\mathbf{I}) p_0(\mathbf{y})\, d\mathbf{y}}{\int_E h(\mathbf{y},\mathbf{x},t)\, d\mathbf{y}} \\ &\leq \frac{p_{\max} D}{p_{\min}(\|\boldsymbol{\pi}(\mathbf{x})-\mathbf{x}\|-\varepsilon)} \frac{\int_{\mathscr{M}\setminus E} \mathscr{N}(\mathbf{y}|\mathbf{x},\sigma_t^2\mathbf{I})\, d\mathbf{y}}{\int_E \mathscr{N}(\mathbf{y}|\mathbf{x},\sigma_t^2\mathbf{I})\, d\mathbf{y}},\end{aligned}$$

which goes to zero as t goes to zero.

Finally, we move to the second term in (C.13). We start with the same steps as with the first term:

$$\frac{\int_{\mathscr{M}\setminus E} g(\mathbf{y},\mathbf{x},t)\, d\mathbf{y}}{\int_{\mathscr{M}} h(\mathbf{y},\mathbf{x},t)\, d\mathbf{y}} = \frac{\frac{\int_{\mathscr{M}\setminus E} g(\mathbf{y},\mathbf{x},t)\, d\mathbf{y}}{\int_E h(\mathbf{y},\mathbf{x},t)\, d\mathbf{y}}}{1+\frac{\int_{\mathscr{M}\setminus E} h(\mathbf{y},\mathbf{x},t)\, d\mathbf{y}}{\int_E h(\mathbf{y},\mathbf{x},t)\, d\mathbf{y}}} =: \frac{C_t}{1+B_t},$$

so we only need to examine C_t. By the triangle and Cauchy-Schwarz inequalities again, we have

$$\begin{aligned}|C_t| &\leq \frac{\int_{\mathscr{M}\setminus E} |\boldsymbol{\nu}^T(\mathbf{y}-\mathbf{x})| \mathscr{N}(\mathbf{y}|\mathbf{x},\sigma_t^2\mathbf{I}) p_0(\mathbf{y})\, d\mathbf{y}}{\int_E h(\mathbf{y},\mathbf{x},t)\, d\mathbf{y}} \\ &\leq \frac{p_{\max} D}{p_{\min}(\|\boldsymbol{\pi}(\mathbf{x})-\mathbf{x}\|-\varepsilon)} \frac{\int_{\mathscr{M}\setminus E} \mathscr{N}(\mathbf{y}|\mathbf{x},\sigma_t^2\mathbf{I})\, d\mathbf{y}}{\int_E \mathscr{N}(\mathbf{y}|\mathbf{x},\sigma_t^2\mathbf{I})\, d\mathbf{y}},\end{aligned}$$

which goes to zero as t goes to zero.

Putting it all together:

$$\begin{aligned}\frac{|\boldsymbol{\nu}^T\nabla_{\mathbf{x}}\ln p_t(\mathbf{x})|}{|\mathbf{n}^T\nabla_{\mathbf{x}}\ln p_t(\mathbf{x})|} &= \frac{|A_t+C_t|}{|1+B_t|} \\ &\leq \frac{1}{|1+B_t|}\left(\frac{p_{\max}}{p_{\min}}\frac{\varepsilon}{\|\boldsymbol{\pi}(\mathbf{x})-\mathbf{x}\|-\varepsilon}+|C_t|\right) \\ &\xrightarrow[t\to 0]{} \frac{p_{\max}}{p_{\min}}\frac{\varepsilon}{\|\boldsymbol{\pi}(\mathbf{x})-\mathbf{x}\|-\varepsilon}.\end{aligned}$$

Since ε can be chosen arbitrarily small, this finishes the proof. □

Proof of Theorem 5.4.1

Beginning with $\mathbf{n}$ and extending to an orthonormal basis $(\mathbf{n}, v_1, \ldots, v_{d-1})$ of $\mathbb{R}^d$, we have

$$\begin{aligned} \mathrm{S}_{\cos}(\mathbf{n}, \nabla_{\mathbf{x}} \ln p_t(\mathbf{x})) &= \frac{\mathbf{n}^T \nabla_{\mathbf{x}} \ln p_t(\mathbf{x})}{\|\nabla_{\mathbf{x}} \ln p_t(\mathbf{x})\|} = \frac{\mathbf{n}^T (\nabla_{\mathbf{x}} \ln p_t(\mathbf{x}))}{\sqrt{(\mathbf{n}^T \nabla_{\mathbf{x}} \ln p_t(\mathbf{x}))^2 + \sum_{i=1}^{d-1} (v_i^T \nabla_{\mathbf{x}} \ln p_t(\mathbf{x}))^2}} \\ &= \frac{1}{\sqrt{1 + \sum_{i=1}^{d-1} \left(\frac{v_i^T \nabla_{\mathbf{x}} \ln p_t(\mathbf{x})}{\mathbf{n}^T \nabla_{\mathbf{x}} \ln p_t(\mathbf{x})}\right)^2}} \xrightarrow[t \to 0]{} 1, \end{aligned}$$

where in taking the limit we applied the Theorem C.4.1. □

Proof of Corollary 5.4.2

Let $(\tau_1, \ldots, \tau_k)$ be an orthonormal basis of the tangent space $T_{\pi(\mathbf{x})}\mathscr{M}$ and extend $\mathbf{n}$ to an orthonormal basis $(\mathbf{n}, \eta_1, \ldots, \eta_{d-k-1})$ of the normal space $\mathscr{N}_{\pi(\mathbf{x})}\mathscr{M}$. Then the projection of the score on the tangent space is given by $\mathbf{T}\nabla_{\mathbf{x}} \ln p_t(\mathbf{x}) = \sum_{i=1}^{k} (\tau_i^T \nabla_{\mathbf{x}} \ln p_t(\mathbf{x}))\tau_i$, while the projection on the normal space is $\mathbf{N}\nabla_{\mathbf{x}} \ln p_t(\mathbf{x}) = (\mathbf{n}^T \nabla_{\mathbf{x}} \ln p_t(\mathbf{x}))\mathbf{n} + \sum_{i=1}^{d-k-1} (\eta_i^T \nabla_{\mathbf{x}} \ln p_t(\mathbf{x}))\eta_i$. Therefore

$$\begin{aligned} \frac{\|\mathbf{T}\nabla_{\mathbf{x}} \ln p_t(\mathbf{x})\|}{\|\mathbf{N}\nabla_{\mathbf{x}} \ln p_t(\mathbf{x})\|} &= \sqrt{\frac{\sum_{i=1}^{k} (\tau_i^T \nabla_{\mathbf{x}} \ln p_t(\mathbf{x}))^2}{(\mathbf{n}^T \nabla_{\mathbf{x}} \ln p_t(\mathbf{x}))^2 + \sum_{i=1}^{d-k-1} (\eta_i^T \nabla_{\mathbf{x}} \ln p_t(\mathbf{x}))^2}} \\ &= \sqrt{\frac{\sum_{i=1}^{k} \left(\frac{\tau_i^T \nabla_{\mathbf{x}} \ln p_t(\mathbf{x})}{\mathbf{n}^T \nabla_{\mathbf{x}} \ln p_t(\mathbf{x})}\right)^2}{1 + \sum_{i=1}^{d-k-1} \left(\frac{\eta_i^T \nabla_{\mathbf{x}} \ln p_t(\mathbf{x})}{\mathbf{n}^T \nabla_{\mathbf{x}} \ln p_t(\mathbf{x})}\right)^2}} \xrightarrow[t \to 0]{} 0 \end{aligned}$$

where in taking the limit we applied the Theorem C.4.1. □

C.5 Details on the design of synthetic image manifolds

Squares image manifold: We generated the *k-squares dataset* whose intrinsic dimension is controllable and is set to k. This dataset comprises 32×32 pixel images of squares, generated by first establishing fixed square center locations and side lengths (either 3 or 5 units) across all datapoints. For each square in a given image, we uniformly sampled a brightness value from the unit interval for all pixels within the square's boundary, summing values at points of intersection. The dimension of this manifold is equal to the number of squares k and the ambient space dimension is $32 \times 32 = 1024$. We provide samples in Figure C.2.

Gaussian blobs image manifold: The Squares image manifold is contained in a low dimensional linear subspace which allowed PPCA to estimate the dimension very well. For this reason, we constructed a synthetic image manifold of known dimension, which cannot be contained in any low dimensional linear subspace. We replace the squares by Gaussian blobs (i.e. brightness of pixels withing each blob is proportional to a Gaussian pdf). We randomly pick the centers of the Gaussian blobs which remain fixed for all datapoints. For each datapoint, we sample the standard deviation of each Gaussian blob uniformly from the interval $[1,5]$. The dimension of this manifold is equal to the number of Gaussian blobs. We provide samples in Figure C.3.

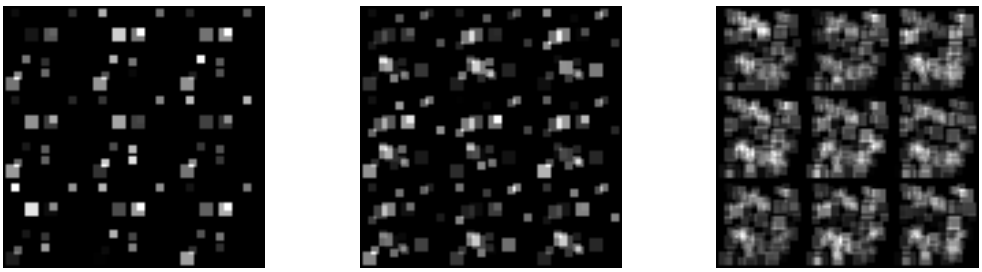


Fig. C.2 Nine samples from Squares image manifolds of dimensions 10, 20 and 100 (from left to right).

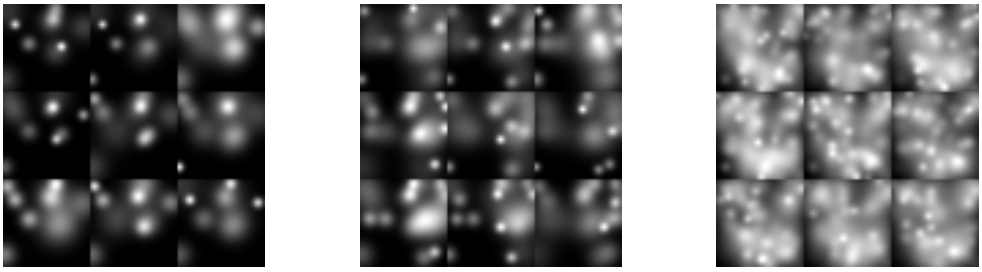


Fig. C.3 Nine samples from Gaussian blob image manifolds of dimensions 10, 20 and 100 (from left to right).

C.6 Additional Experimental Results

C.6.1 Euclidean Data

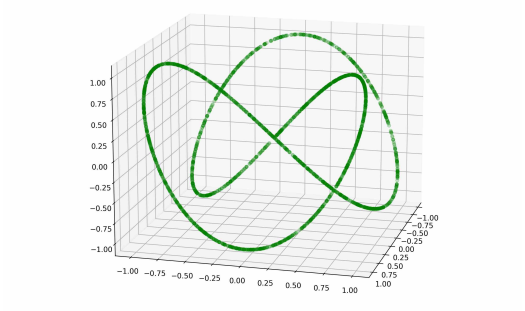


Fig. C.4 Projection of the spaghetti line on the first three dimensions.

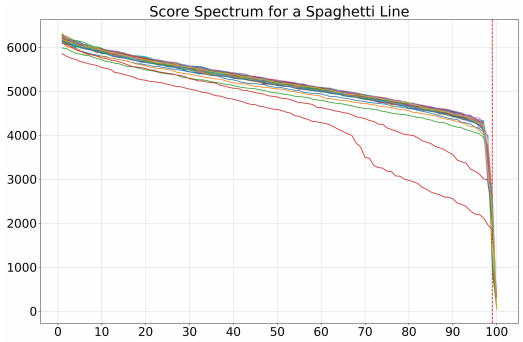


Fig. C.5 Score spectrum of the spaghetti line. The last singular value clearly vanishes indicating that the intrinsic dimensionality of the manifold is equal to one.

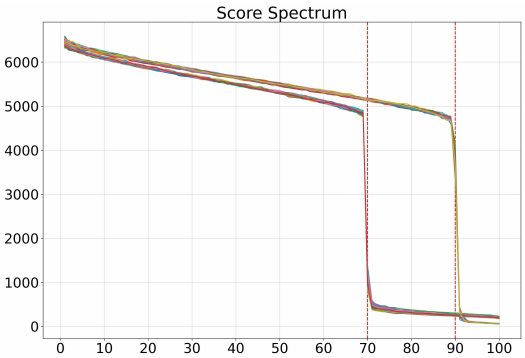


Fig. C.6 Score spectrum for the union of k-spheres ($k_1 = 10, k_2 = 30$). The separated drops in the spectra clearly show that the data comes form the union of two manifolds of different dimensions.

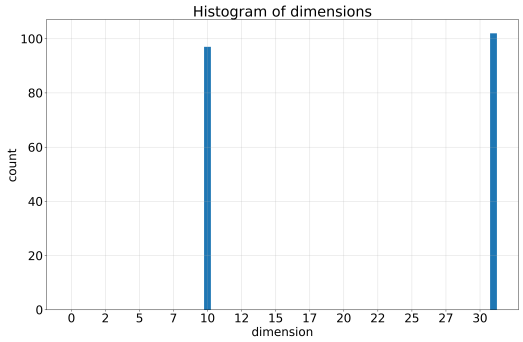


Fig. C.7 The histogram of estimated dimensions for the union of k-spheres ($k_1 = 10, k_2 = 30$). The counts are taken over estimates $\hat{k}(\mathbf{x}_0^{(i)})$ at different points $\mathbf{x}_0^{(i)}$.

C.6.2 Synthetic Image Data

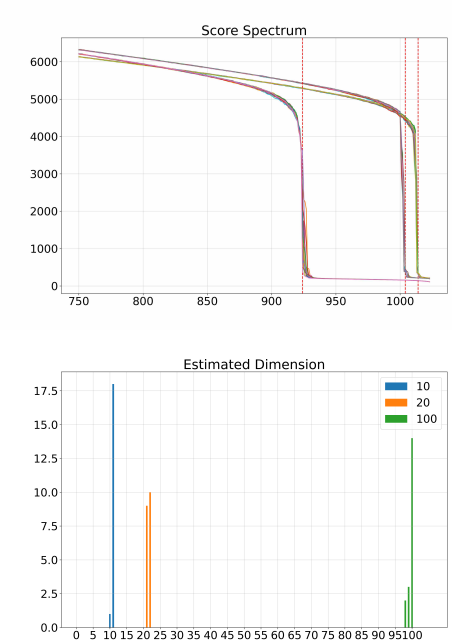


Fig. C.8 Score spectra and histogram of estimated dimension based on the score spectrum of the Squares image manifold of dimensions 10, 20 and 100.

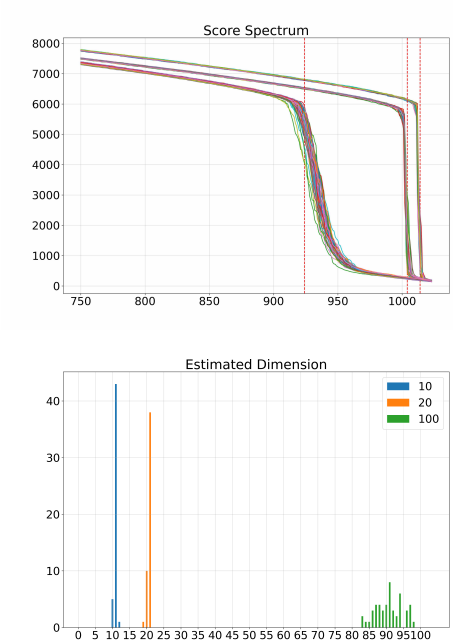


Fig. C.9 Score spectra and histogram of estimated dimension based on the score spectrum of the Gaussian blobs image manifold of dimensions 10, 20 and 100.

C.6.3 MNIST

In Figure C.10 we present 500 score spectra evaluated at 500 different instances for each digit. We observe that a considerable number of spectra indicate lower manifold dimension than the dimension documented in Table 5.1. This deviation can be attributed to amplified geometrical and statistical error at the respective evaluation points.

We choose the maximum estimated dimension as our conjecture of the intrinsic dimension under the premise that a collapse of the spectrum at a smaller dimension than the dimension of the normal space is extremely unlikely from a probabilistic standpoint. However, it is feasible to encounter a spectrum collapse suggesting a higher normal space dimension, hence a lower intrinsic dimension, due to the intensified geometric and approximation error at the point of evaluation. It is noteworthy that our estimation strategy locates the drop at the position of the maximal gradient, a practice that may marginally inflate the intrinsic dimension estimate, as exemplified in the Squares and Gaussian blobs manifolds. Therefore, if our reported

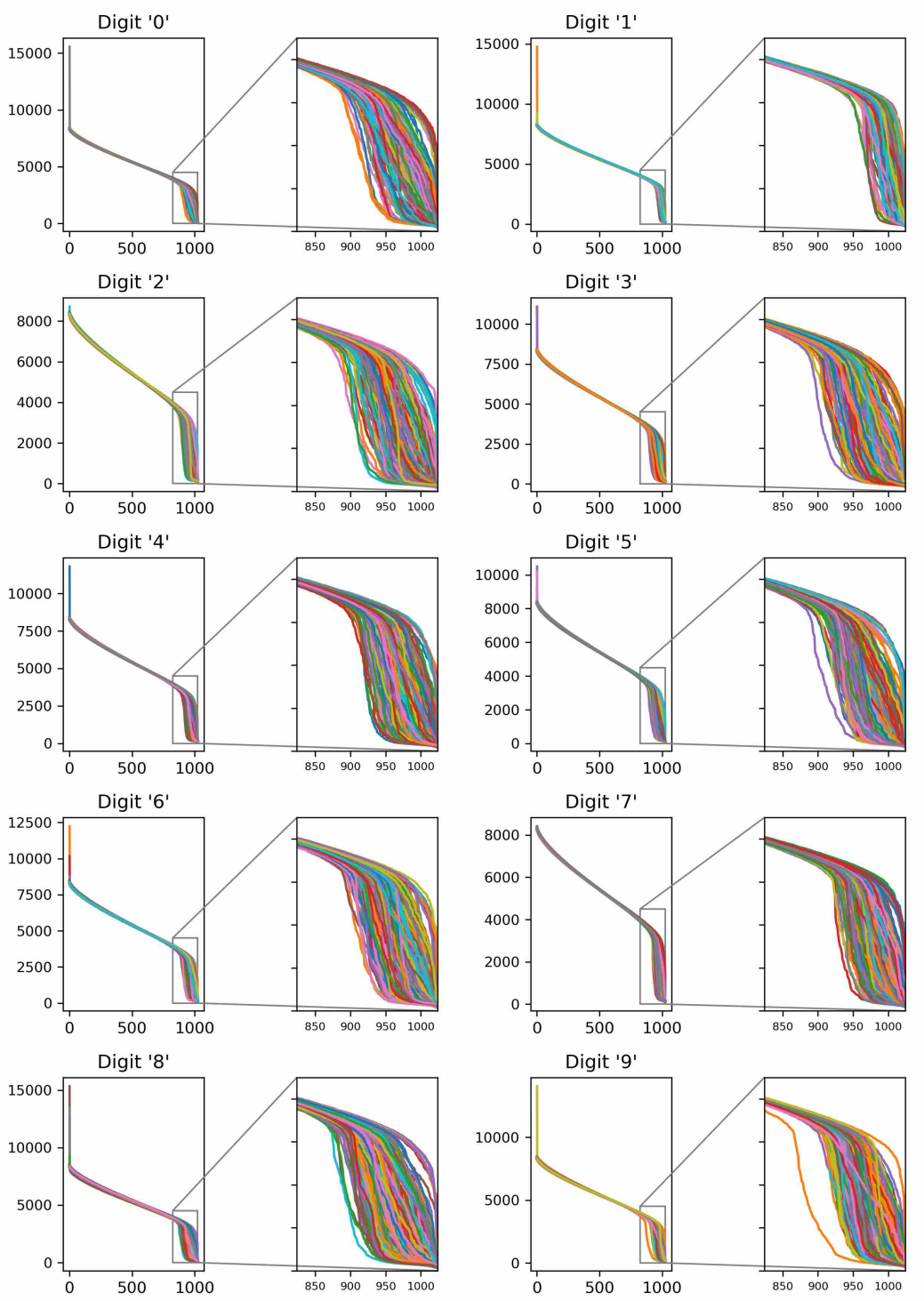


Fig. C.10 MNIST score spectra for all digits

dimension for each MNIST digit is not precise, it is either a minor overestimation or a lower limit of the true dimension.

C.7 Robustness analysis

C.7.1 Robustness to score approximation error

As we discussed in the previous sections, our method is guaranteed to work given a perfect score approximation for sufficiently small t. However, in practice there will be an approximation error resulting from finite training data, limited model capacity and imperfect optimization. Therefore, we conduct an empirical analysis of the influence of the error in score approximation on the produced estimate of the dimension. We train a model s_θ on a uniform distribution on 25-sphere and then we corrupt the output of the model with a Gaussian perturbation $e \sim \mathcal{N}(0, \sigma_e^2 \mathbf{I})$. Then, we produce score spectra by applying our method to the resulting corrupted scores. We repeat this experiment for different intensities of noise σ_e. We pick σ_e so that the noise norm to score norm ratio $r = \mathbb{E}[\|e\|]/\mathbb{E}_{x_{t_0} \sim p_{t_0}(\mathbf{x}_{t_0}|\mathbf{x}_0)}[\|s(\mathbf{x}_{t_0}, t_0)\|]$ has a prescribed value. We find that as we increase the intensity of noise singular values corresponding to the tangential component start to increase causing the gap in the score spectrum to diminish. This is expected since the noise added to the score vectors has a tangential component. However, for values of $r < 0.5$ our method is still producing a visible drop in the spectrum at the correct point. The results are presented in Figure C.11.

C.7.2 Robustness to non-uniform distribution on the manifold

	Uniform	$\alpha = 1$	$\alpha = 0.75$	$\alpha = 0.5$
Ground Truth	10	10	10	10
Ours	11	10	10	7
MLE (m=5)	9.61	5.37	4.83	4.12
MLE (m=20)	9.46	4.99	4.49	3.82
Local PCA	11	5	4	3
PPCA	11	11	11	11

Table C.2 Dimensionality detection for non-uniform distribution. For our method the maximum over pointwise estimates $\hat{k}(\mathbf{x}_0)$ is considered.

We examine the robustness to our method to non-unifomity in data distribution on the manifold surface. Under perfect score approximation and sufficiently small t_0 our method is guaranteed to work, but we conduct an empirical study to investigate the behaviour in the presence of score approximation error and $t_0 > 0$ used in practice in diffusion models. We consider a k-sphere and sample the surface of the sphere in a non-uniform fashion. We obtain points on the k-sphere by sampling vectors $\boldsymbol{\theta}$ of $k-1$ angles (in radians) from a

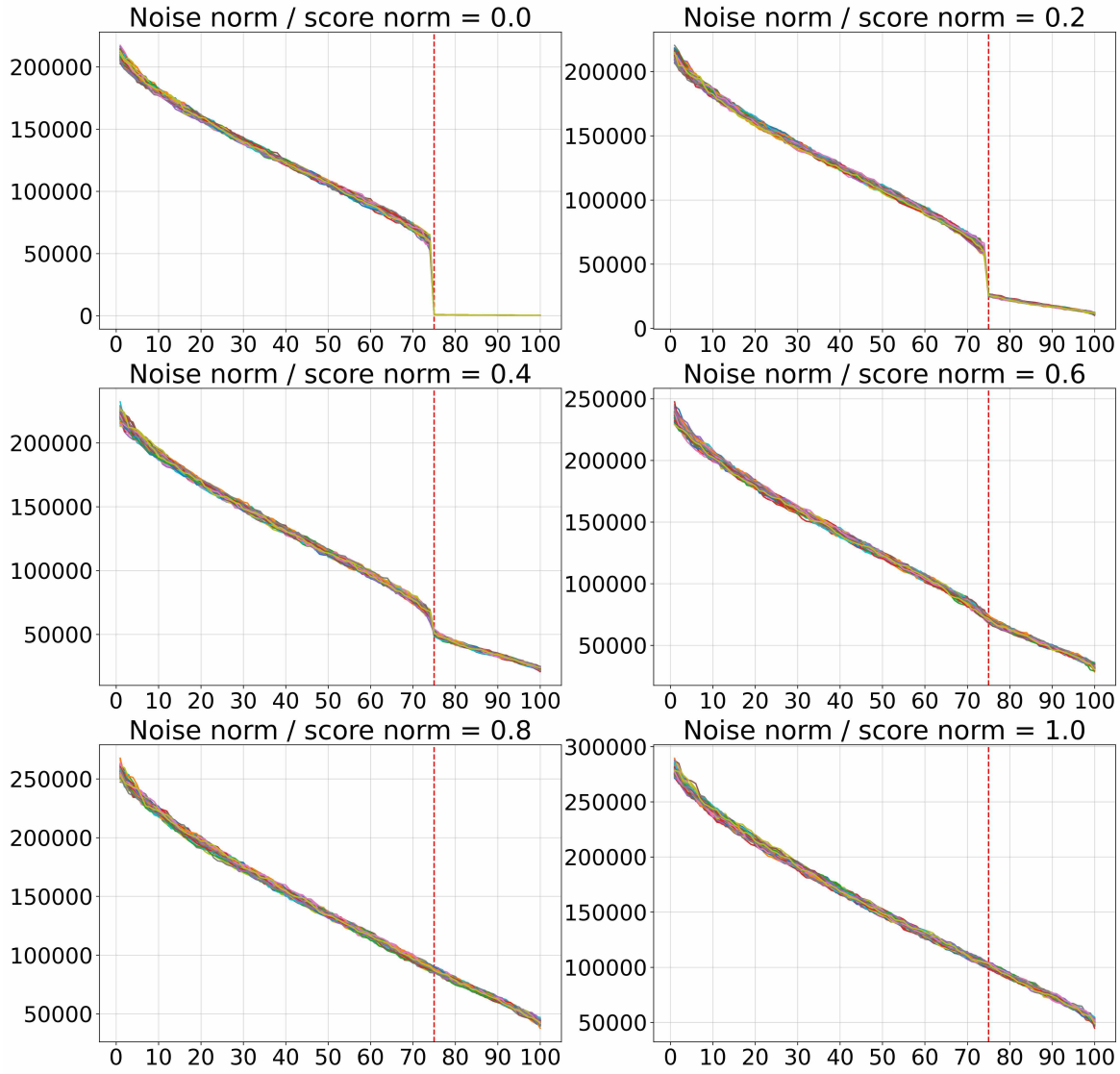


Fig. C.11 Score spectra for noise corrupted score model on 25-sphere.

Gaussian distribution $\mathcal{N}(0, \alpha \mathbf{I})$, where $\alpha \in \mathbb{R}$ is a constant that determines the degree of non-uniformity. Then, we embed the resulting points via a random isometry in a 100 dimensional ambient space. We sample $n = 1000$ points $\mathbf{x}_0^{(j)}$ from the manifold and at each point we estimate the dimensionality $\hat{k}(\mathbf{x}_0^{(j)})$ via the score spectrum. The pointwise estimates are presented in Figure C.12 and final estimates are shown in Table C.2. For values of $\alpha \in \{1, 0.75\}$, we can still obtain an accurate estimate of the dimension if we take $\hat{k} = \max_{j=1,\ldots,1000} \hat{k}(\mathbf{x}_0^{(j)})$ the maximum over point-wise estimates. For an extremely concentrated distribution with $\alpha = 0.5$ the method underestimates the dimension, which indicates that the tangential component of the score was not approximately constant in the neighborhoods used to sample the scores. This problem could be further mitigated by ap-

proximating the score closer to the manifold and using smaller sampling neighborhoods (i.e. for smaller t_0) or sampling more points $\mathbf{x}_0^{(j)}$. Notice that taking the maximum over $\hat{k}(\mathbf{x}_0^{(j)})$ is theoretically justified (as long as we assume we are dealing with a connected manifold) since our method can underestimate due to geometric or approximation error (cf. sections 5.4 and 5.5, but it is unlikely to significantly overestimate.

C.7.3 Relaxing the strict manifold assumption

The proof of the Theorem 5.4.1 assumes that p_0 is strictly contained on the data manifold $\mathscr{M}$, however in practice it is possible that the data distribution is concentrated around $\mathscr{M}$ rather then being contained within. Therefore, we conduct an empirical analysis, which examines how our method works in the case of the data contained around the manifold. We start with p_0 a uniform distribution on a unit 25-sphere embedded in a 100 dimensional space and convolve it with a Gaussian distribution $\mathscr{N}(0, \sigma\mathbf{I})$ to obtain a distribution p_0^σ that concentrates around $\mathscr{M}$. As σ increases the distribution is blurred out more and more into the ambient space. We train score model on each of the resulting distributions and use our method to estimate the dimension. We find that our method produces correct estimation for for small values of σ i.e. when p_0^σ is still concentrated tightly around $\mathscr{M}$. This is expected since of high values of σ the distribution isn't really concentrated around any manifold and therefore the notion of intrinsic dimension doesn't make any sense. The results are presented in Figure C.13.

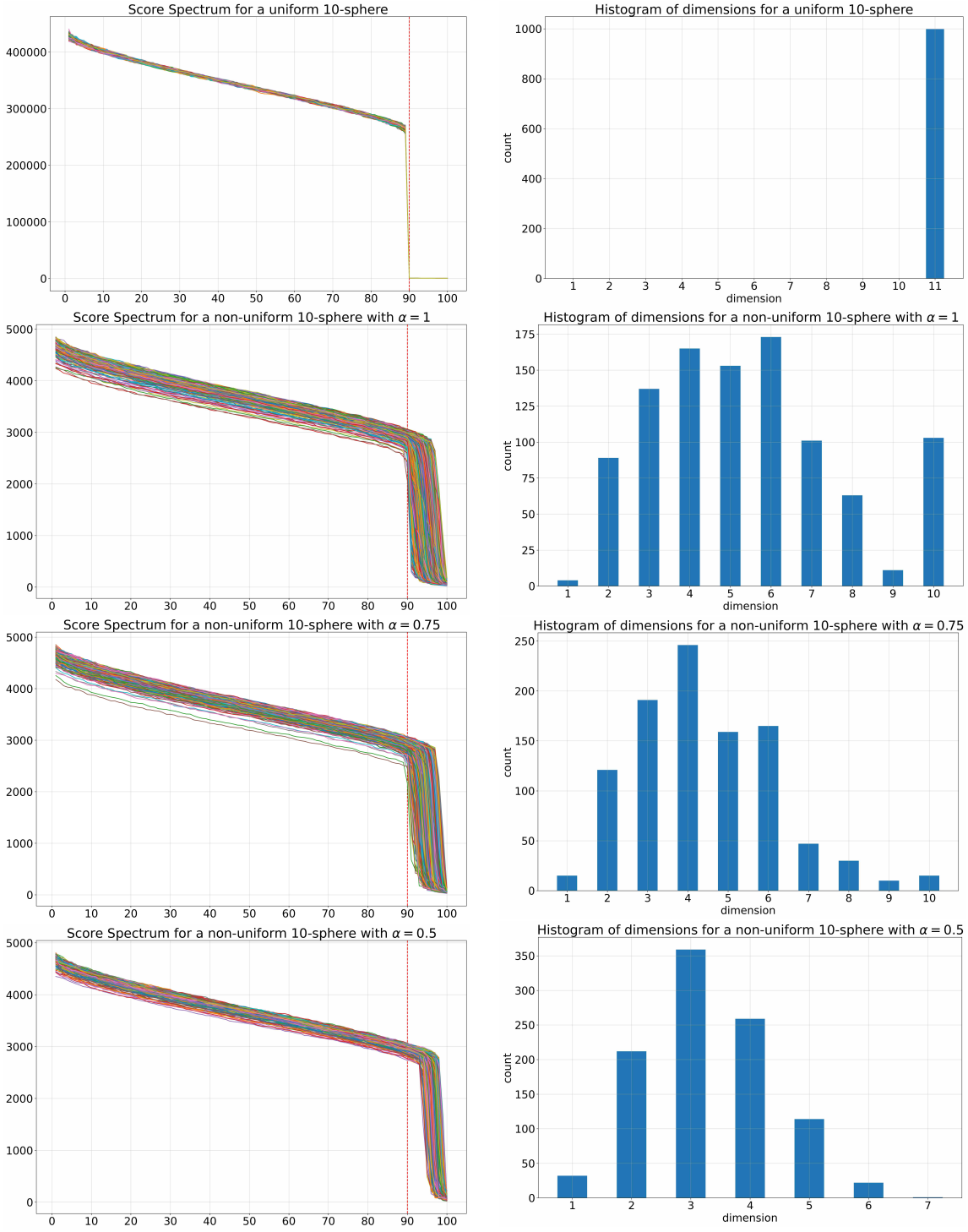


Fig. C.12 Dimensionality estimates for uniform and non-uniform distributions on a 10-sphere. On the right, we present histograms showing how many points $\mathbf{x}_0^{(j)}$ result in a given $\hat{k}(\mathbf{x}_0^{(j)})$. Taking $\hat{k} = \max_j \hat{k}(\mathbf{x}_0^{(j)})$ allows for robust estimation for moderate values of the concentration parameter α.

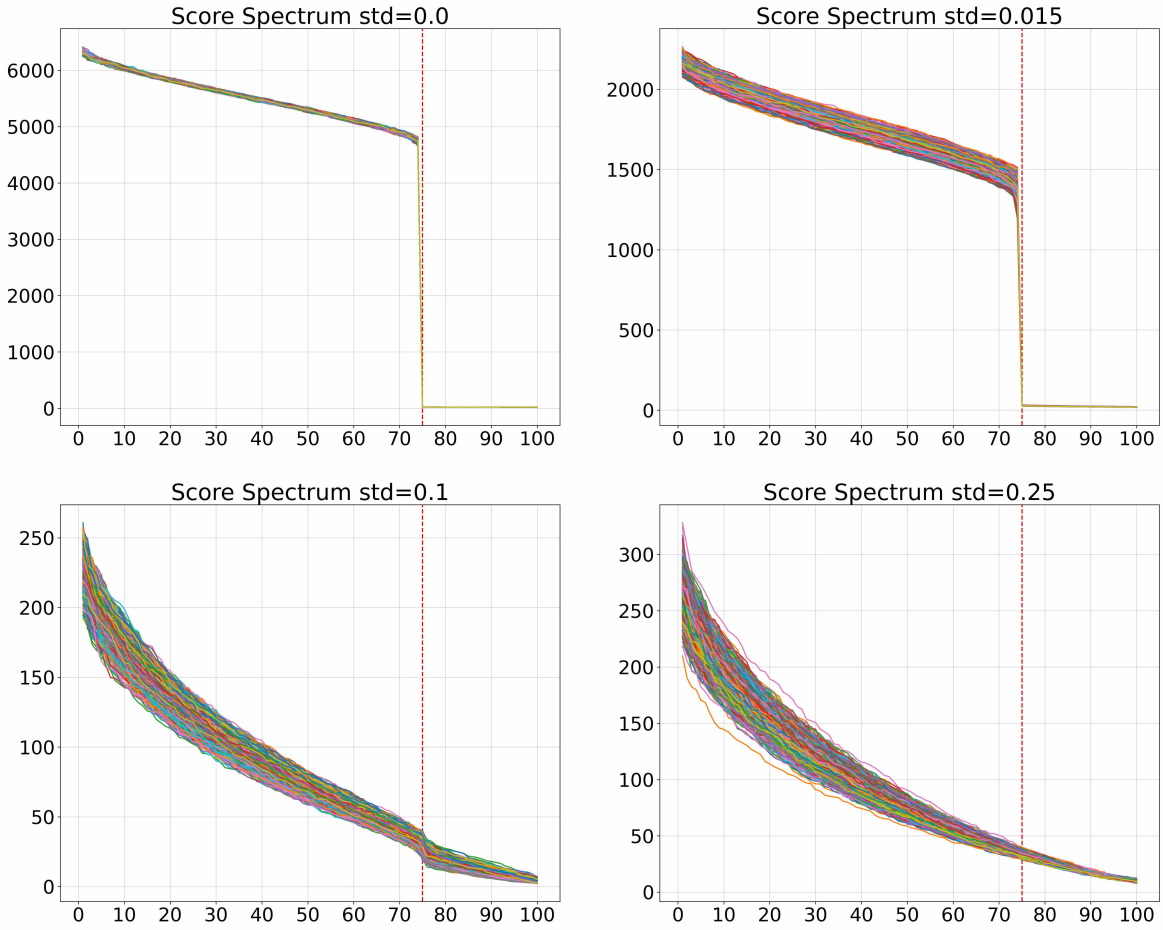


Fig. C.13 Score spectra for score models on 25-sphere trained on noisy manifold data.

Appendix D

Variational Diffusion Auto-encoder

D.0.1 Full derivation of the training objective

In this section we derive the maximum likelihood training objective for the encoder network. Let $s_\theta(\mathbf{x}_t, t)$ be a score function of a pre-trained unconditional diffusion model and let $e_\phi : (\mathbf{x}_t, t) \mapsto (\mu^z_\phi(\mathbf{x}_t, t), \sigma^z_\phi(\mathbf{x}_t, t)I)$ be the encoder network. The neural approximation of the conditional data score is given by

$$s_{\theta,\phi}(\mathbf{x}_t, \mathbf{z}, t) := s_\theta(\mathbf{x}_t, t) + \nabla_{\mathbf{x}_t} \ln q_{t,\phi}(\mathbf{z}|\mathbf{x}_t)$$

where $q_{t,\phi}(\mathbf{z}|\mathbf{x}_t) = \mathcal{N}\big(\mathbf{z}; \mu^z_\phi(\mathbf{x}_t, t), \sigma^z_\phi(\mathbf{x}_t, t)I\big)$.

Recall that by variational lower bound, for any $\mathbf{x}_0, \mathbf{z}$ and distribution $q(\mathbf{z}|\mathbf{x}_0)$ we have

$$\ln p_{\theta,\phi}(\mathbf{x}_0) \geq \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x}_0)}[\ln p_{\theta,\phi}(\mathbf{x}_0|\mathbf{z})] - D_{KL}(q(\mathbf{z}|\mathbf{x}_0) \parallel p(\mathbf{z})) \tag{D.1}$$

Moreover, by [181, Theorem 3] for any $\mathbf{x}_0$ and $\mathbf{z}$ we have

$$\ln p_{\theta,\phi}(\mathbf{x}_0|\mathbf{z}) \geq \mathscr{L}_{DSM}(\mathbf{x}_0, \mathbf{z}) \tag{D.2}$$

where

$$\begin{aligned}\mathscr{L}_{DSM}(\mathbf{x}_0, \mathbf{z}) := \mathbb{E}_{\mathbf{x}_t\sim p_t(\mathbf{x}_t|\mathbf{x}_0)}[\ln \pi(\mathbf{x}_t)] - \mathbb{E}_{\substack{t\sim U(0,T)\\ \mathbf{x}_t\sim p_t(\mathbf{x}_t|\mathbf{x}_0)}}\big[g(t)^2 \big\|\nabla_{\mathbf{x}_t} \ln p_t(\mathbf{x}_t|\mathbf{x}_0) - s_{\theta,\phi}(\mathbf{x}_t, \mathbf{z}, t)\big\|_2^2\big] \\ + \mathbb{E}_{\substack{t\sim U(0,T)\\ \mathbf{x}_t\sim p_t(\mathbf{x}_t|\mathbf{x}_0)}}\big[g(t)^2 \|\nabla_{\mathbf{x}_t} \ln p_t(\mathbf{x}_t|\mathbf{x}_0)\|_2^2 + 2\nabla_{\mathbf{x}_t} \cdot \mathbf{f}(\mathbf{x}_t, t)\big]\end{aligned}$$

Putting both together we obtain:

$$\begin{aligned}\ln p_{\theta,\phi}(\mathbf{x}_0) &\geq \mathbb{E}_{\mathbf{z}\sim q_{0,\phi}(\mathbf{z}|\mathbf{x}_0)}[\ln p_{\theta,\phi}(\mathbf{x}_0|\mathbf{z})] - D_{KL}(q_{0,\phi}(\mathbf{z}|\mathbf{x}_0) \parallel p(\mathbf{z})) \\ &\geq \mathbb{E}_{\mathbf{z}\sim q_{0,\phi}(\mathbf{z}|\mathbf{x}_0)}[\mathscr{L}_{DSM}(\mathbf{x}_0,\mathbf{z})] - D_{KL}(q_{0,\phi}(\mathbf{z}|\mathbf{x}_0) \parallel p(\mathbf{z}))\end{aligned}$$

after removing terms which don't depend on the parameters of the model and taking average over the data-points, we obtain the following training objective

$$\mathscr{L}(\phi) := \mathbb{E}_{\mathbf{x}_0\sim p(\mathbf{x}_0)}\Bigg[\frac{1}{2}\mathbb{E}_{\substack{t\sim U(0,T)\\ \mathbf{x}_t\sim p_t(\mathbf{x}_t|\mathbf{x}_0)\\ \mathbf{z}\sim q_{0,\phi}(\mathbf{z}|\mathbf{x}_t)}}\left[g(t)^2 \left\|\nabla_{\mathbf{x}_t}\ln p_t(\mathbf{x}_t|\mathbf{x}_0) - s_{\theta,\phi}(\mathbf{x}_t,\mathbf{z},t)\right\|_2^2\right] + D_{KL}\big(q_{0,\phi}(\mathbf{z}|\mathbf{x}_0) \parallel p(\mathbf{z})\big)\Bigg]$$

It follows from the above considerations that

$$\arg\min_{\phi} \mathscr{L}(\phi) = \arg\max_{\phi} \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}[\ln p_{\theta,\phi}(\mathbf{x})]$$

or in other words that minimizing the objective $\mathscr{L}(\phi)$ is equivalent to maximizing the marginal data log-likelihood $\ln p_{\theta,\phi}(\mathbf{x})$.

Finally just like in β-VAE we find that in practice it is helpful to introduce a hyper-parameter $\beta \in [0,1]$ which controls the strength of KL regularization. We define our final training objective as:

$$\mathscr{L}_{\beta}(\phi) := \mathbb{E}_{\mathbf{x}_0\sim p(\mathbf{x}_0)}\Bigg[\frac{1}{2}\mathbb{E}_{\substack{t\sim U(0,T)\\ \mathbf{x}_t\sim p_t(\mathbf{x}_t|\mathbf{x}_0)\\ \mathbf{z}\sim q_{0,\phi}(\mathbf{z}|\mathbf{x}_t)}}\left[g(t)^2 \left\|\nabla_{\mathbf{x}_t}\ln p_t(\mathbf{x}_t|\mathbf{x}_0) - s_{\theta,\phi}(\mathbf{x}_t,\mathbf{z},t)\right\|_2^2\right] + \beta D_{KL}\big(q_{0,\phi}(\mathbf{z}|\mathbf{x}_0) \parallel p(\mathbf{z})\big)\Bigg]$$

D.0.2 Pixel-wise L_2 is not a perceptual metric

As we discussed in the previous section choosing a Gaussian model for $p(\mathbf{x}|\mathbf{z})$ in VAEs leads to a term in the loss function, which is equivalent to the L_2 reconstruction error. While L_2 is a common dissimilarity metric it is a very bad choice for certain data modalities such as images. This is because L_2 distance measures the differences between corresponding pixel intensities, but does not take into account human perception. Thus, two images may have a small L_2 distance but still appear visually different, or vice versa (see Figure D.1). The metric does not consider the hierarchical and contextual information that humans use when

perceiving images. In particular small spatial shifts, rotations or cropping can lead to large L_2 distances even if the images are perceptually similar.

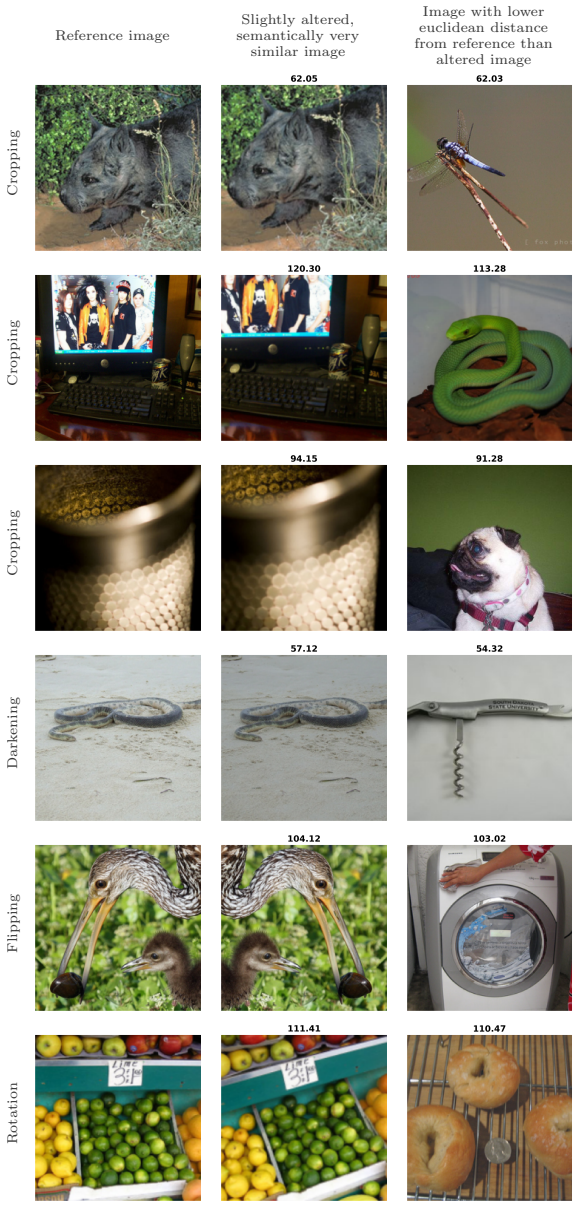


Fig. D.1 Examples where the L_2-distance is not a (semantically) meaningful distance between images. In the left column, a reference image from ImageNet [39] is shown. In the middle column, a slight alteration is applied, whose result a human observer would consider to be very close to the reference image. In the right column, another image from the ImageNet data set is displayed, which to the human observer is very different from the reference image, but which has a lower L_2-distance to the reference image than the altered image. The numbers above the images indicate the L_2-distance to the reference image. Figure taken from [189].

D.1 Experimental details

D.1.1 ScoreVAE

The pretrained diffusion models for all experiments are based on the DDPM architecture [78]. We used 128 base filters and attention at resolution 16×16 for all experiments. For Cifar10, we set the channel multiplier array to $[1,2,2,2]$ and the number of ResNet blocks to 4. For CelebA 64×64, we set the channel multiplier array to $[1,1,2,2,3]$ and the number of ResNet blocks to 2. We used 0.1 dropout rate for Cifar10 and 0 dropout rate for CelebA. We used the beta-linear variance preserving forward process with the same parameters as the ones used by [183] and trained the diffusion model using the weighted denoising score matching objective with the simple weighting, i.e., $\lambda(t) = \sigma(t)^2$, where $\sigma(t)$ is the standard deviation of the perturbation kernel. We used the Adam optimizer and EMA rate 0.999. Finally, we set the learning rate to $2e{-}4$ for Cifar10 and $1e{-}4$ for CelebA.

The time dependent encoder for the Cifar10 experiment is a simple convolutional network that consists of a sequence of blocks of convolutions followed by the GELU activation function. The final activation is flattened and concatenated to the time tensor. A final linear layer maps the time augmented flattened tensor to the latent dimension. The time dependent encoder for CelebA is based heavily on the DDPM architecture. We removed the upsampling part of the U-NET and removed the skip connections. The downscaled tensor is flattened and mapped to the latent dimension with an additional linear layer. We used the Adam optimizer and EMA rate 0.999. We set the learning rate to $2e{-}4$ for Cifar10 and $1e{-}4$ for CelebA. We trained the Cifar10 encoder for $1.4M$ iterations and the CelebA encoder for $300K$ iterations.

D.1.2 VAE

For VAE we used exactly the same encoder architectures as in the Score VAE (except they were not conditioned on time). For each choice of encoder we created a mirror decoder with symmetric architecture. In Cifar10 the decoder starts by reshaping the the flat latent vector into a tensor which is then passed through a sequence of transposed convolutions which exactly mirror the structure of the encoder. In CelebA we used a decoder consisting of the upsampling part of the DDPM U-NET. The Cifar10 model was trained for 11M iterations, while the CelebA model was trained for 600K iterations.

D.2 Extended qualitative evaluation

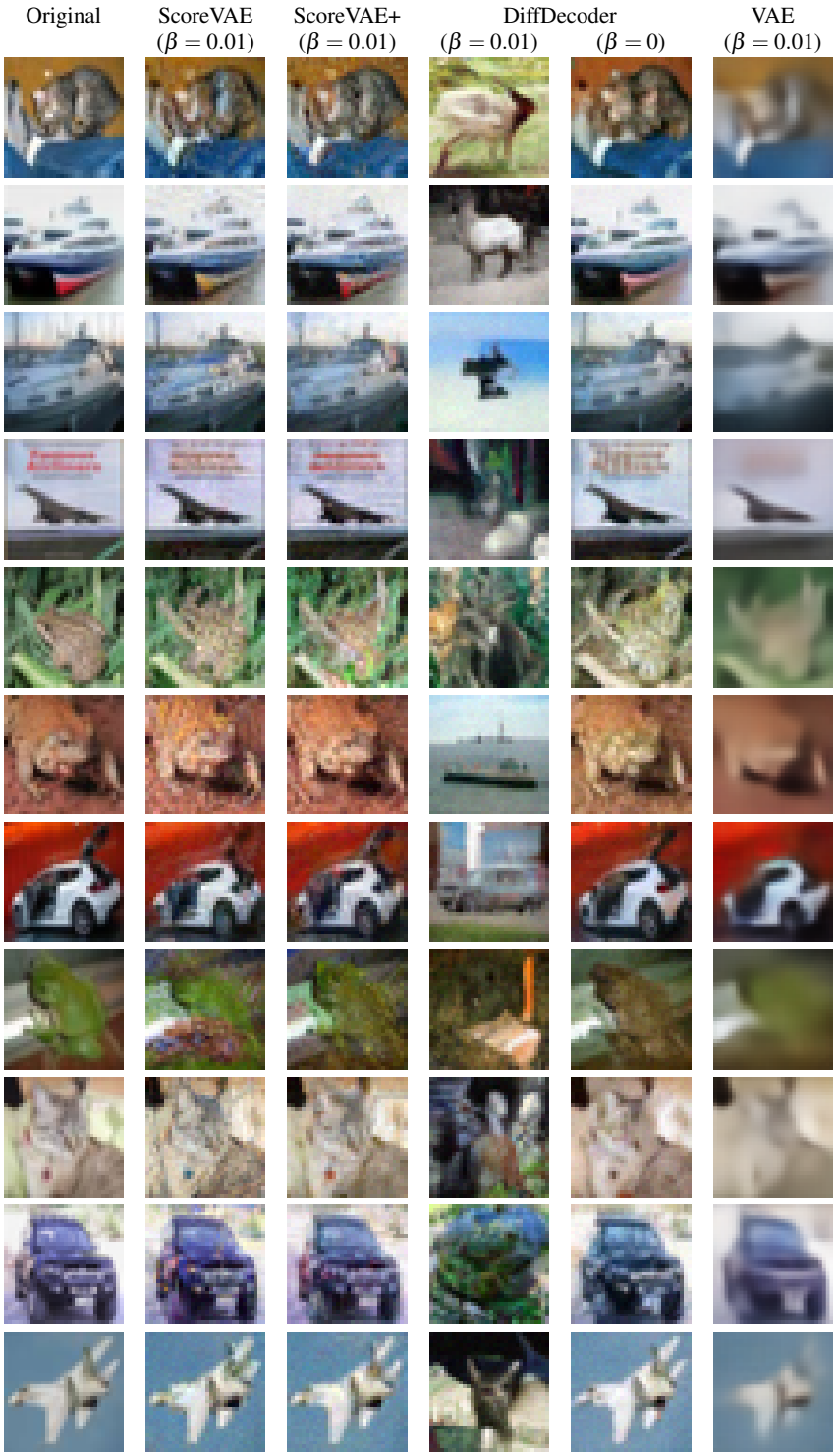


Table D.1 Cifar10

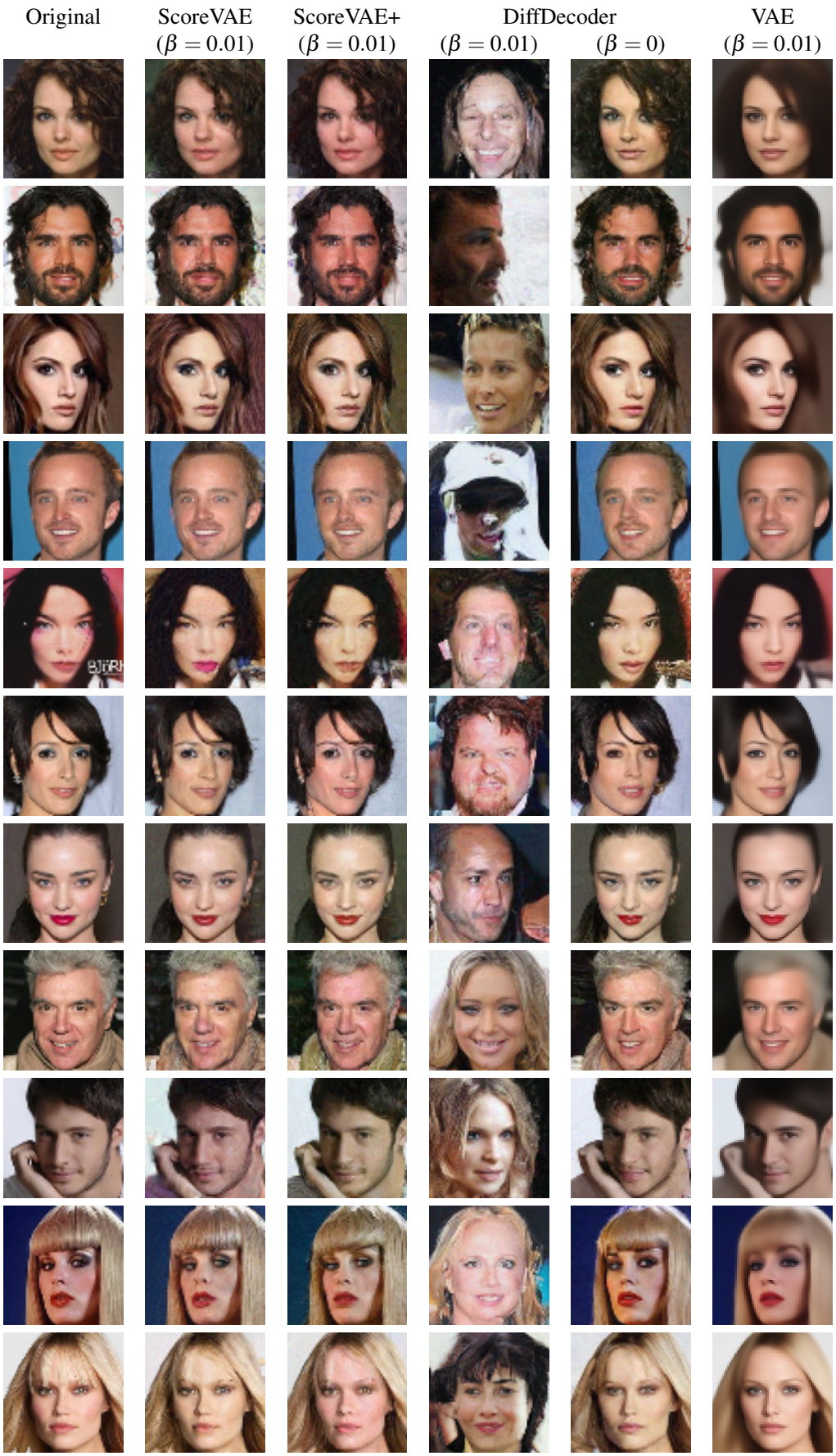


Table D.2 CelebA

Appendix E

Score-based pullback Riemannian geometry

E.1 Proof of 7.4.1

Auxiliary lemma

Lemma E.1.1. *Let $\varphi : \mathbb{R}^d \to \mathbb{R}^d$ be a smooth diffeomorphism and let $\psi : \mathbb{R}^d \to \mathbb{R}$ be a quadratic function of the form 7.6 with diagonal $\mathbf{A} \in \mathbb{R}^{d\times d}$. Furthermore, let $p : \mathbb{R}^d \to \mathbb{R}$ be the corresponding probability density of the form 7.3. Finally, consider $\varepsilon \in [0,1]$ and the mappings $E_\varepsilon : \mathbb{R}^d \to \mathbb{R}^{d_\varepsilon}$ and $D_\varepsilon : \mathbb{R}^{d_\varepsilon} \to \mathbb{R}^d$ in 7.20,7.21 with $d_\varepsilon \in [d]$ as in 7.19.*

Then, for any $\alpha \in [0,1)$ and any $\beta \in [0, 1-\alpha)$

$$\mathbb{E}_{\mathbf{X}\sim p}[d^{\varphi}_{\mathbb{R}^d}(D_\varepsilon(E_\varepsilon(\mathbf{X})),\mathbf{X})^2 e^{\frac{\alpha}{2}\varphi(\mathbf{X})^\top \mathbf{A}^{-1}\varphi(\mathbf{X})}] \leq \varepsilon \frac{C^2_{\beta,\varphi} C^3_{\beta,\varphi}}{1-\alpha-\beta}\Big(\frac{1+\beta}{1-\alpha-\beta}\Big)^{\frac{d}{2}} \sum_{i=1}^{d} \mathbf{a}_i, \tag{E.1}$$

where

$$C^3_{\beta,\varphi} := \sup_{\mathbf{x}\in\mathbb{R}^d} \{|\det(D_{\varphi(\mathbf{x})}\varphi^{-1})| e^{-\frac{\beta}{2}\varphi(\mathbf{x})^\top \mathbf{A}^{-1}\varphi(\mathbf{x})}\}, \tag{E.2}$$

and

$$C^2_{\beta,\varphi} := \sup_{\mathbf{x}\in\mathbb{R}^d} \{|\det(D_{\mathbf{x}}\varphi)| e^{-\frac{\beta}{2}\varphi(\mathbf{x})^\top \mathbf{A}^{-1}\varphi(\mathbf{x})}\}. \tag{E.3}$$

Proof. We need to distinct two cases: (i) $d_\varepsilon = d$ and (ii) $1 \leq d_\varepsilon < d$

(i) If $d_\varepsilon = d$ we have that $D_\varepsilon(E_\varepsilon(\mathbf{x})) = \mathbf{x}$ for any $\mathbf{x} \in \mathbb{R}^d$. In other words

$$\mathbb{E}_{\mathbf{X}\sim p}[d^{\varphi}_{\mathbb{R}^d}(D_\varepsilon(E_\varepsilon(\mathbf{X})),\mathbf{X})^2 e^{\frac{\alpha}{2}\varphi(\mathbf{X})^\top \mathbf{A}^{-1}\varphi(\mathbf{X})}] = 0 \leq \varepsilon \frac{C^2_{\beta,\varphi} C^3_{\beta,\varphi}}{1-\alpha-\beta}\Big(\frac{1+\beta}{1-\alpha-\beta}\Big)^{\frac{d}{2}} \sum_{i=1}^{d} \mathbf{a}_i. \tag{E.4}$$

(ii) Next, we consider the case $1 \leq d_\varepsilon < d$. First, notice that we can rewrite

$$\|\varphi(D_\varepsilon(E_\varepsilon(\mathbf{x}))) - \varphi(\mathbf{x})\|_2^2 \overset{??}{=} \|\sum_{k=1}^{d_\varepsilon}(\varphi(\mathbf{x}), \mathbf{e}^{i_k})_2 \mathbf{e}^{i_k} - \varphi(\mathbf{x})\|_2^2 = \|\sum_{k=d_\varepsilon+1}^{d}(\varphi(\mathbf{x}), \mathbf{e}^{i_k})_2 \mathbf{e}^{i_k}\|_2^2$$

$$\overset{\text{orthogonality}}{=} \sum_{k=d_\varepsilon+1}^{d} \|(\varphi(\mathbf{x}), \mathbf{e}^{i_k})_2 \mathbf{e}^{i_k}\|_2^2 = \sum_{k=d_\varepsilon+1}^{d}(\varphi(\mathbf{x}), \mathbf{e}^{i_k})_2^2 = \sum_{k=d_\varepsilon+1}^{d} \varphi(\mathbf{x})_{i_k}^2. \quad \text{(E.5)}$$

Moreover, we define

$$C := \int_{\mathbb{R}^d} e^{-\frac{1}{2}\varphi(\mathbf{x})^\top \mathbf{A}^{-1}\varphi(\mathbf{x})} \mathrm{d}\mathbf{x}. \quad \text{(E.6)}$$

Then,

$$\mathbb{E}_{\mathbf{X}\sim p}[d^{\varphi}_{\mathbb{R}^d}(D_\varepsilon(E_\varepsilon(\mathbf{X})),\mathbf{X})^2 e^{\frac{\alpha}{2}\varphi(\mathbf{X})^\top \mathbf{A}^{-1}\varphi(\mathbf{X})}] = \frac{\int_{\mathbb{R}^d} \|\varphi(D_\varepsilon(E_\varepsilon(\mathbf{x}))) - \varphi(\mathbf{x})\|_2^2 e^{-(\frac{1}{2}-\frac{\alpha}{2})\varphi(\mathbf{x})^\top \mathbf{A}^{-1}\varphi(\mathbf{x})}\mathrm{d}\mathbf{x}}{\int_{\mathbb{R}^d} e^{-\frac{1}{2}\varphi(\mathbf{x})^\top \mathbf{A}^{-1}\varphi(\mathbf{x})}\mathrm{d}\mathbf{x}}$$

$$\overset{\text{E.6}}{=} \frac{1}{C}\int_{\mathbb{R}^d} \|\varphi(D_\varepsilon(E_\varepsilon(\mathbf{x}))) - \varphi(\mathbf{x})\|_2^2 e^{-(\frac{1}{2}-\frac{\alpha}{2})\varphi(\mathbf{x})^\top \mathbf{A}^{-1}\varphi(\mathbf{x})}\mathrm{d}\mathbf{x}$$

$$\overset{\text{E.5}}{=} \frac{1}{C}\int_{\mathbb{R}^d} \sum_{k=d_\varepsilon+1}^{d} \varphi(\mathbf{x})_{i_k}^2 e^{-(\frac{1}{2}-\frac{\alpha}{2})\varphi(\mathbf{x})^\top \mathbf{A}^{-1}\varphi(\mathbf{x})}\mathrm{d}\mathbf{x} = \frac{1}{C}\sum_{k=d_\varepsilon+1}^{d}\int_{\mathbb{R}^d} \varphi(\mathbf{x})_{i_k}^2 e^{-(\frac{1}{2}-\frac{\alpha}{2})\varphi(\mathbf{x})^\top \mathbf{A}^{-1}\varphi(\mathbf{x})}\mathrm{d}\mathbf{x}$$

$$\overset{\mathbf{x}=\varphi^{-1}(\mathbf{y})}{=} \frac{1}{C}\sum_{k=d_\varepsilon+1}^{d}\int_{\mathbb{R}^d} \mathbf{y}_{i_k}^2 e^{-(\frac{1}{2}-\frac{\alpha}{2})\mathbf{y}^\top \mathbf{A}^{-1}\mathbf{y}} |\det(D_\mathbf{y}\varphi^{-1})|\mathrm{d}\mathbf{y}$$

$$= \frac{1}{C}\sum_{k=d_\varepsilon+1}^{d}\int_{\mathbb{R}^d} \mathbf{y}_{i_k}^2 e^{-(\frac{1}{2}-\frac{\alpha}{2}-\frac{\beta}{2})\mathbf{y}^\top \mathbf{A}^{-1}\mathbf{y}} |\det(D_\mathbf{y}\varphi^{-1})| e^{-\frac{\beta}{2}\mathbf{y}^\top \mathbf{A}^{-1}\mathbf{y}}\mathrm{d}\mathbf{y}$$

$$\leq \frac{\sup_{\mathbf{y}\in\mathbb{R}^d}\{|\det(D_\mathbf{y}\varphi^{-1})| e^{-\frac{\beta}{2}\mathbf{y}^\top \mathbf{A}^{-1}\mathbf{y}}\}}{C}\sum_{k=d_\varepsilon+1}^{d}\int_{\mathbb{R}^d} \mathbf{y}_{i_k}^2 e^{-(\frac{1}{2}-\frac{\alpha}{2}-\frac{\beta}{2})\mathbf{y}^\top \mathbf{A}^{-1}\mathbf{y}}\mathrm{d}\mathbf{y}$$

$$\overset{\text{E.2}}{=} \frac{C_{\beta,\varphi}^2}{C}\sum_{k=d_\varepsilon+1}^{d}\int_{\mathbb{R}^d} \mathbf{y}_{i_k}^2 e^{-(\frac{1}{2}-\frac{\alpha}{2}-\frac{\beta}{2})\mathbf{y}^\top \mathbf{A}^{-1}\mathbf{y}}\mathrm{d}\mathbf{y} = \frac{C_{\beta,\varphi}^2}{C}\sum_{k=d_\varepsilon+1}^{d}\int_{\mathbb{R}^d} \mathbf{y}_{i_k}^2 e^{-(\frac{1}{2}-\frac{\alpha}{2}-\frac{\beta}{2})\sum_{j=1}^{d}\frac{\mathbf{y}_j^2}{\mathbf{a}_j}}\mathrm{d}\mathbf{y}$$

$$= \frac{C_{\beta,\varphi}^2}{C}\sum_{k=d_\varepsilon+1}^{d}\int_{\mathbb{R}} \mathbf{y}_{i_k}^2 e^{-(\frac{1}{2}-\frac{\alpha}{2}-\frac{\beta}{2})\frac{\mathbf{y}^2}{\mathbf{a}_{i_k}}}\mathrm{d}\mathbf{y}_{i_k}\int_{\mathbb{R}^{d-1}} e^{-(\frac{1}{2}-\frac{\alpha}{2}-\frac{\beta}{2})\sum_{j\neq i_k}^{d}\frac{\mathbf{y}_j^2}{\mathbf{a}_j}}\mathrm{d}\mathbf{y}_1\ldots\mathrm{d}\mathbf{y}_{i_k-1}\mathrm{d}\mathbf{y}_{i_k+1}\ldots\mathrm{d}\mathbf{y}_d$$

$$= \frac{C_{\beta,\varphi}^2}{C}\sum_{k=d_\varepsilon+1}^{d}\frac{\mathbf{a}_{i_k}}{(1-\alpha-\beta)}\int_{\mathbb{R}} e^{-(\frac{1}{2}-\frac{\alpha}{2}-\frac{\beta}{2})\frac{\mathbf{y}^2}{\mathbf{a}_{i_k}}}\mathrm{d}\mathbf{y}_{i_k}\int_{\mathbb{R}^{d-1}} e^{-(\frac{1}{2}-\frac{\alpha}{2}-\frac{\beta}{2})\sum_{j\neq i_k}^{d}\frac{\mathbf{y}_j^2}{\mathbf{a}_j}}\mathrm{d}\mathbf{y}_1\ldots\mathrm{d}\mathbf{y}_{i_k-1}\mathrm{d}\mathbf{y}_{i_k+1}\ldots\mathrm{d}\mathbf{y}_d$$

$$= \frac{C_{\beta,\varphi}^2}{C}\sum_{k=d_\varepsilon+1}^{d}\frac{\mathbf{a}_{i_k}}{(1-\alpha-\beta)}\int_{\mathbb{R}^d} e^{-(\frac{1}{2}-\frac{\alpha}{2}-\frac{\beta}{2})\mathbf{y}^\top \mathbf{A}^{-1}\mathbf{y}}\mathrm{d}\mathbf{y}$$

$$= \frac{C_{\beta,\varphi}^2}{C}\sum_{k=d_\varepsilon+1}^{d}\frac{\mathbf{a}_{i_k}}{(1-\alpha-\beta)}\Big(\frac{1+\beta}{1-\alpha-\beta}\Big)^{\frac{d}{2}}\int_{\mathbb{R}^d} e^{-(\frac{1}{2}+\frac{\beta}{2})\mathbf{y}^\top \mathbf{A}^{-1}\mathbf{y}}\mathrm{d}\mathbf{y}$$

$$\overset{\mathbf{y}=\varphi(\mathbf{x})}{=} \frac{C_{\beta,\varphi}^2}{C}\sum_{k=d_\varepsilon+1}^{d}\frac{\mathbf{a}_{i_k}}{(1-\alpha-\beta)}\Big(\frac{1+\beta}{1-\alpha-\beta}\Big)^{\frac{d}{2}}\int_{\mathbb{R}^d} e^{-(\frac{1}{2}+\frac{\beta}{2})\varphi(\mathbf{x})^\top \mathbf{A}^{-1}\varphi(\mathbf{x})}|\det(D_\mathbf{x}\varphi)|\mathrm{d}\mathbf{x}$$

$$= \frac{C_{\beta,\varphi}^2}{C}\sum_{k=d_\varepsilon+1}^{d}\frac{\mathbf{a}_{i_k}}{(1-\alpha-\beta)}\Big(\frac{1+\beta}{1-\alpha-\beta}\Big)^{\frac{d}{2}}\int_{\mathbb{R}^d} e^{-\frac{1}{2}\varphi(\mathbf{x})^\top \mathbf{A}^{-1}\varphi(\mathbf{x})}|\det(D_\mathbf{x}\varphi)| e^{-\frac{\beta}{2}\varphi(\mathbf{x})^\top \mathbf{A}^{-1}\varphi(\mathbf{x})}\mathrm{d}\mathbf{x}$$

$$\leq \frac{C_{\beta,\varphi}^2 \sup_{\mathbf{x}\in\mathbb{R}^d}\{|\det(D_\mathbf{x}\varphi)| e^{-\frac{\beta}{2}\varphi(\mathbf{x})^\top \mathbf{A}^{-1}\varphi(\mathbf{x})}\}}{C}\sum_{k=d_\varepsilon+1}^{d}\frac{\mathbf{a}_{i_k}}{(1-\alpha-\beta)}\Big(\frac{1+\beta}{1-\alpha-\beta}\Big)^{\frac{d}{2}}\int_{\mathbb{R}^d} e^{-\frac{1}{2}\varphi(\mathbf{x})^\top \mathbf{A}^{-1}\varphi(\mathbf{x})}\mathrm{d}\mathbf{x}$$

$$\overset{\text{E.3}}{=} \frac{C_{\beta,\varphi}^2 C_{\beta,\varphi}^3}{C}\sum_{k=d_\varepsilon+1}^{d}\frac{\mathbf{a}_{i_k}}{(1-\alpha-\beta)}\Big(\frac{1+\beta}{1-\alpha-\beta}\Big)^{\frac{d}{2}}\int_{\mathbb{R}^d} e^{-\frac{1}{2}\varphi(\mathbf{x})^\top \mathbf{A}^{-1}\varphi(\mathbf{x})}\mathrm{d}\mathbf{x}$$

$$\overset{\text{E.6}}{=} \frac{C_{\beta,\varphi}^2 C_{\beta,\varphi}^3}{1-\alpha-\beta}\Big(\frac{1+\beta}{1-\alpha-\beta}\Big)^{\frac{d}{2}}\sum_{k=d_\varepsilon+1}^{d}\mathbf{a}_{i_k}$$

$$\overset{7.19}{\leq} \varepsilon\frac{C_{\beta,\varphi}^2 C_{\beta,\varphi}^3}{1-\alpha-\beta}\Big(\frac{1+\beta}{1-\alpha-\beta}\Big)^{\frac{d}{2}}\sum_{i=1}^{d}\mathbf{a}_i. \quad \text{(E.7)}$$

□

Proof of the theorem

Proof of 7.4.1. First, consider the Taylor approximation

$$\begin{aligned}\varphi^{-1}(\varphi(\mathbf{y}))-\varphi^{-1}(\varphi(\mathbf{y}))&=D_{\varphi(\mathbf{x})}\varphi^{-1}[\varphi(\mathbf{y})-\varphi(\mathbf{x})]+\mathscr{O}(\|\varphi(\mathbf{y})-\varphi(\mathbf{x})\|_2^2)\\&=D_{\varphi(\mathbf{x})}\varphi^{-1}[\varphi(\mathbf{y})-\varphi(\mathbf{x})]+\mathscr{O}(d^{\varphi}_{\mathbb{R}^d}(\mathbf{y},\mathbf{x})^2).\end{aligned}\tag{E.8}$$

Moreover, we define

$$C:=\int_{\mathbb{R}^d}e^{-\frac{1}{2}\varphi(\mathbf{x})^\top\mathbf{A}^{-1}\varphi(\mathbf{x})}\mathrm{d}\mathbf{x}.\tag{E.9}$$

Subsequently, notice that

$$\begin{aligned}&\mathbb{E}_{\mathbf{X}\sim p}[\|D_{\varphi(\mathbf{X})}\varphi^{-1}[\varphi(D_\varepsilon(E_\varepsilon(\mathbf{X})))-\varphi(\mathbf{X})]\|_2^2]\\&\quad=\frac{1}{C}\int_{\mathbb{R}^d}\|D_{\varphi(\mathbf{x})}\varphi^{-1}[\varphi(D_\varepsilon(E_\varepsilon(\mathbf{x})))-\varphi(\mathbf{x})]\|_2^2e^{-\frac{1}{2}\varphi(\mathbf{x})^\top\mathbf{A}^{-1}\varphi(\mathbf{x})}\mathrm{d}\mathbf{x}\\&\quad\le\frac{1}{C}\int_{\mathbb{R}^d}\|D_{\varphi(\mathbf{x})}\varphi^{-1}\|_2^2\|\varphi(D_\varepsilon(E_\varepsilon(\mathbf{x})))-\varphi(\mathbf{x})\|_2^2e^{-\frac{1}{2}\varphi(\mathbf{x})^\top\mathbf{A}^{-1}\varphi(\mathbf{x})}\mathrm{d}\mathbf{x}\\&\le\frac{\sup_{\mathbf{x}\in\mathbb{R}^d}\{\|D_{\varphi(\mathbf{x})}\varphi^{-1}\|_2^2e^{-\frac{\beta}{2}\varphi(\mathbf{x})^\top\mathbf{A}^{-1}\varphi(\mathbf{x})}\}}{C}\int_{\mathbb{R}^d}\|\varphi(D_\varepsilon(E_\varepsilon(\mathbf{x})))-\varphi(\mathbf{x})\|_2^2e^{-(\frac{1}{2}-\frac{\beta}{2})\varphi(\mathbf{x})^\top\mathbf{A}^{-1}\varphi(\mathbf{x})}\mathrm{d}\mathbf{x}\\&\overset{7.23}{=}\frac{C^1_{\beta,\varphi}}{C}\int_{\mathbb{R}^d}\|\varphi(D_\varepsilon(E_\varepsilon(\mathbf{x})))-\varphi(\mathbf{x})\|_2^2e^{\frac{\beta}{2}\varphi(\mathbf{x})^\top\mathbf{A}^{-1}\varphi(\mathbf{x})}e^{-\frac{1}{2}\varphi(\mathbf{x})^\top\mathbf{A}^{-1}\varphi(\mathbf{x})}\mathrm{d}\mathbf{x}\\&\quad=C^1_{\beta,\varphi}\mathbb{E}_{\mathbf{X}\sim p}[d^{\varphi}_{\mathbb{R}^d}(D_\varepsilon(E_\varepsilon(\mathbf{X})),\mathbf{X})^2e^{\frac{\beta}{2}\varphi(\mathbf{X})^\top\mathbf{A}^{-1}\varphi(\mathbf{X})}]\\&\quad\overset{\text{E.1.1}}{\le}\varepsilon\frac{C^1_{\beta,\varphi}C^2_{\beta,\varphi}C^3_{\beta,\varphi}}{1-2\beta}\Big(\frac{1+\beta}{1-2\beta}\Big)^{\frac{d}{2}}\sum_{i=1}^d\mathbf{a}_i.\end{aligned}\tag{E.10}$$

Then,

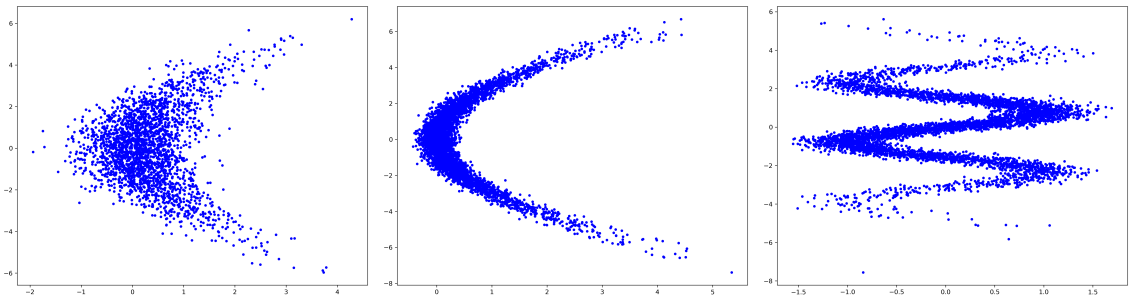
$$\begin{aligned}\mathbb{E}_{\mathbf{X}\sim p}[\|D_\varepsilon(E_\varepsilon(\mathbf{X}))-\mathbf{X}\|_2^2]&=\mathbb{E}_{\mathbf{X}\sim p}[\|\varphi^{-1}(\varphi(D_\varepsilon(E_\varepsilon(\mathbf{X}))))-\varphi^{-1}(\varphi(\mathbf{X}))\|_2^2]\\&\overset{E.8}{=}\mathbb{E}_{\mathbf{X}\sim p}[\|D_{\varphi(\mathbf{X})}\varphi^{-1}[\varphi(D_\varepsilon(E_\varepsilon(\mathbf{X})))-\varphi(\mathbf{X})]+\mathscr{O}(d^{\varphi}_{\mathbb{R}^d}(D_\varepsilon(E_\varepsilon(\mathbf{X})),\mathbf{X})^2)\|_2^2]\\&=\mathbb{E}_{\mathbf{X}\sim p}[\|D_{\varphi(\mathbf{X})}\varphi^{-1}[\varphi(D_\varepsilon(E_\varepsilon(\mathbf{X})))-\varphi(\mathbf{X})]\|_2^2+\mathscr{O}(d^{\varphi}_{\mathbb{R}^d}(D_\varepsilon(E_\varepsilon(\mathbf{X})),\mathbf{X})^3)]\\&\overset{\text{E.10}}{\le}\varepsilon\frac{C^1_{\beta,\varphi}C^2_{\beta,\varphi}C^3_{\beta,\varphi}}{1-2\beta}\Big(\frac{1+\beta}{1-2\beta}\Big)^{\frac{d}{2}}\sum_{i=1}^d\mathbf{a}_i+o(\varepsilon),\end{aligned}\tag{E.11}$$

which yields the claim as β was arbitrary. □

E.2 Dataset Construction Details

In this section, we provide a detailed explanation of the construction of the datasets used in our experiments. We organize the datasets into two categories based on the experimental sections in which they are used.

E.2.1 Datasets for Manifold Mapping Experiments



(a) Single Banana (b) Squeezed Single Banana (c) River

Fig. E.1 Visualization of the datasets used in our manifold mapping experiments.

In our manifold mapping experiments (7.6.1), we use the following datasets illustrated in E.1:

- *Single Banana Dataset*: A two-dimensional dataset shaped like a curved banana.
- *Squeezed Single Banana Dataset*: A variant of the Single Banana with a tighter bend.
- *River Dataset*: A more complex 2D dataset resembling the meandering path of a river.

Each dataset is constructed by defining specific diffeomorphisms φ and convex quadratic functions ψ, then sampling from the resulting probability density using Langevin Monte Carlo Markov Chain (MCMC) with Metropolis-Hastings correction. The probability density function is defined as:

$$p(\mathbf{x}) \propto e^{-\psi(\varphi(\mathbf{x}))}, \tag{E.12}$$

where the strongly convex function ψ is given by:

$$\psi(\mathbf{v}) = \frac{1}{2}\mathbf{v}^\top A^{-1}\mathbf{v}, \tag{E.13}$$

and A is a positive-definite diagonal matrix. The specific choices of φ and A for each dataset determine its geometric properties.

Diffeomorphisms and Convex Quadratic Functions

The key differences between the datasets arise from the diffeomorphism φ and the covariance matrix $\mathbf{A}$ used in the sampling process. Below, we describe the specific settings for each dataset.

1. Single Banana Dataset

- Diffeomorphism:
$$\varphi(\mathbf{x}) = \begin{pmatrix} x_1 - ax_2^2 - z \\ x_2 \end{pmatrix}$$
where $a = \frac{1}{9}$ and $z = 0$.
- Covariance matrix:
$$\mathbf{A} = \begin{pmatrix} \frac{1}{4} & 0 \\ 0 & 4 \end{pmatrix}$$

2. Squeezed Single Banana Dataset

- Diffeomorphism: Same as the Single Banana Dataset.
- Covariance matrix:
$$\mathbf{A} = \begin{pmatrix} \frac{1}{81} & 0 \\ 0 & 4 \end{pmatrix}$$

3. River Dataset

- Diffeomorphism:
$$\varphi(\mathbf{x}) = \begin{pmatrix} x_1 - \sin(ax_2) - z \\ x_2 \end{pmatrix}$$
where $a = 2$ and $z = 0$.
- Covariance matrix:
$$\mathbf{A} = \begin{pmatrix} \frac{1}{25} & 0 \\ 0 & 3 \end{pmatrix}$$

Dataset Generation Algorithm

Algorithm 2 outlines the dataset generation process for all three datasets. The specific diffeomorphisms and quadratic functions differ for each dataset.

Algorithm 2 General Dataset Generation Algorithm

Require: Number of samples N, MCMC steps T, Step size δ, Diffeomorphism φ, Covariance matrix ■

Ensure: Dataset $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$

Initialize: Set initial state $\mathbf{x}_0 = \mathbf{0} \in \mathbb{R}^2$.
for $i = 1$ to N **do**
  $\mathbf{x} = \mathbf{x}_0$
  for $k = 1$ to T **do**
    Compute the score function $\nabla_{\mathbf{x}} \log p_{\text{target}}(\mathbf{x})$.
    Propose $\mathbf{x}' = \mathbf{x} + \frac{\delta^2}{2}\nabla_{\mathbf{x}} \log p_{\text{target}}(\mathbf{x}) + \delta\eta$, where $\eta \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_2)$.
    Compute the forward kernel:

$$K_{\text{forward}} = \frac{|\mathbf{x} - \mathbf{x}' + \frac{\delta^2}{2}\nabla_{\mathbf{x}'} \log p_{\text{target}}(\mathbf{x}')|^2}{2\delta^2}$$

    Compute the reverse kernel:

$$K_{\text{reverse}} = \frac{|\mathbf{x}' - \mathbf{x} + \frac{\delta^2}{2}\nabla_{\mathbf{x}} \log p_{\text{target}}(\mathbf{x})|^2}{2\delta^2}$$

    Compute the Metropolis-Hastings acceptance probability:

$$A = \min\left(1, \frac{p_{\text{target}}(\mathbf{x}')}{p_{\text{target}}(\mathbf{x})} \exp\left(-K_{\text{forward}} + K_{\text{reverse}}\right)\right)$$

    Accept $\mathbf{x}'$ with probability A; else set $\mathbf{x}' = \mathbf{x}$.
    Update $\mathbf{x} = \mathbf{x}'$.
  end for
  Store the final $\mathbf{x}$ as sample $\mathbf{x}_i$.
end for

E.2.2 Datasets for Riemannian Autoencoder Experiments

In the Riemannian autoencoder experiments (7.6.2), we use the following datasets:

- *Hemisphere*(*d'*, *d*) Dataset: Samples drawn from the upper hemisphere of a *d'*-dimensional unit sphere and embedded into $\mathbb{R}^d$ via a random isometric mapping.
- *Sinusoid*(*d'*, *d*) Dataset: Generated by applying sinusoidal transformations to *d'*-dimensional latent variables, resulting in a complex, nonlinear manifold in $\mathbb{R}^d$.

E.2.3 Hemisphere(d', d) Dataset

The *Hemisphere*(d', d) dataset consists of samples drawn from the upper hemisphere of a d'-dimensional unit sphere, which are then embedded into a d-dimensional ambient space using a random isometric embedding. Below are the steps involved in constructing this dataset.

1. Sampling from the Upper Hemisphere We begin by sampling points from the upper hemisphere of the d'-dimensional unit sphere $S_+^{d'} \subset \mathbb{R}^{d'+1}$. The upper hemisphere is defined as:

$$S_+^{d'} = \left\{ \mathbf{x} \in \mathbb{R}^{d'+1} : \|\mathbf{x}\| = 1, x_1 \geq 0 \right\}.$$

The first angular coordinate θ_1 is sampled from a Beta distribution with shape parameters $\alpha = 5$ and $\beta = 5$, scaled to the interval $\left[0, \frac{\pi}{2}\right]$. This sampling method emphasizes points near the "equator" of the hemisphere. The remaining angular coordinates $\theta_2, \ldots, \theta_{d'}$ are sampled uniformly from the interval $[0, \pi]$:

$$\theta_1 \sim \text{Beta}(5,5) \cdot \left(\frac{\pi}{2}\right), \quad \theta_i \sim \text{Uniform}(0, \pi), \text{ for } i = 2, \ldots, d'.$$

2. Conversion to Cartesian Coordinates Next, each sampled point in spherical coordinates is converted into Cartesian coordinates in $\mathbb{R}^{d'+1}$ using the following transformation equations:

$$x_1 = \cos(\theta_1), \quad x_2 = \sin(\theta_1)\cos(\theta_2), \quad \ldots, \quad x_{d'+1} = \sin(\theta_1)\sin(\theta_2)\cdots\sin(\theta_{d'}).$$

This conversion ensures that the sampled points lie on the surface of the unit sphere in $(d'+1)$-dimensional space.

3. Random Isometric Embedding into $\mathbb{R}^d$ After sampling points on the hemisphere in $\mathbb{R}^{d'+1}$, the points are embedded into a d-dimensional ambient space ($d \geq d'+1$) using a random isometric embedding. The embedding process is as follows:

1. Generate a random matrix $\mathbf{A} \in \mathbb{R}^{d \times (d'+1)}$, where each entry is sampled from a standard normal distribution $\mathcal{N}(0,1)$.

2. Perform a QR decomposition on matrix $\mathbf{A}$ to obtain $\mathbf{Q} \in \mathbb{R}^{d \times (d'+1)}$:

$$\mathbf{A} = \mathbf{QR}.$$

The columns of $\mathbf{Q}$ form an orthonormal basis for a $(d'+1)$-dimensional subspace of $\mathbb{R}^d$, ensuring that $\mathbf{Q}$ defines an isometric embedding from $\mathbb{R}^{d'+1}$ into $\mathbb{R}^d$. This guarantees that distances and angles are preserved during the mapping, maintaining the geometric structure of the original space within the higher-dimensional ambient space.

3. Use matrix $\mathbf{Q}$ to map each sample $\mathbf{x} \in \mathbb{R}^{d'+1}$ into the ambient space:

$$\mathbf{y} = \mathbf{Q}\mathbf{x},$$

where $\mathbf{y} \in \mathbb{R}^d$ are the embedded samples.

E.2.4 Sinusoid(d', d) Dataset

The *Sinusoid*(d', d) dataset represents a d'-dimensional manifold embedded in d-dimensional space through nonlinear sinusoidal transformations. Below are the detailed steps involved in constructing this dataset.

1. Sampling Latent Variables The latent variables $\mathbf{z} \in \mathbb{R}^{d'}$ are sampled from a multivariate Gaussian distribution with zero mean and isotropic variance, as follows:

$$\mathbf{z} \sim \mathcal{N}\left(0, \sigma_m^2 I_{d'}\right),$$

where σ_m^2 controls the variance along each intrinsic dimension, and $I_{d'}$ is the $d' \times d'$ identity matrix. The value of σ_m^2 is set to 3 for our experiments.

2. Defining Ambient Coordinates with Sinusoidal Transformations For each of the $d - d'$ ambient dimensions, we construct a shear vector $\mathbf{a}_j \in \mathbb{R}^{d'}$, with its elements drawn uniformly from the interval $[1,2]$:

$$\mathbf{a}_j \sim \text{Uniform}(1,2)^{d'}, \quad \text{for } j = 1, \ldots, d - d'.$$

The shear vectors $\mathbf{a}_j$ apply a fixed linear transformation to the latent space $\mathbf{z} \in \mathbb{R}^{d'}$, determining how the latent variables influence each ambient dimension. These vectors, sampled once for each of the $d - d'$ ambient dimensions, modulate the scale and periodicity of the sinusoidal transformation.

Each ambient coordinate x_j is generated as a sinusoidal function of the inner product between $\mathbf{a}_j$ and $\mathbf{z}$, with a small Gaussian noise added for regularization.

Algorithm 3 Hemisphere(d', d) Dataset Generation

Input: Intrinsic dimension d', ambient dimension d, number of samples n, Beta distribution parameters $\alpha = 5$, $\beta = 5$
Output: Dataset $\mathbf{Y} \in \mathbb{R}^{n \times d}$
Step 1: Generate Random Isometric Embedding
Generate a random matrix $\mathbf{A} \in \mathbb{R}^{d \times (d'+1)}$ with entries from $\mathcal{N}(0,1)$
Perform QR decomposition on $\mathbf{A}$ to obtain $\mathbf{Q} \in \mathbb{R}^{d \times (d'+1)}$:

$$\mathbf{A} = \mathbf{QR}$$

Step 2: Construct Dataset
for $i = 1$ to n **do**
    Step 2.1: Sample Spherical Coordinates
    Sample the first angular coordinate θ_1 from a scaled Beta distribution:

$$\theta_1 \sim \text{Beta}(\alpha, \beta) \cdot \left(\frac{\pi}{2}\right)$$

    Sample the remaining angular coordinates $\theta_2, \ldots, \theta_{d'}$ from a uniform distribution:

$$\theta_i \sim \text{Uniform}(0, \pi), \quad \text{for } i = 2, \ldots, d'$$

    Step 2.2: Convert to Cartesian Coordinates
    Convert the spherical coordinates to Cartesian coordinates $\mathbf{x}_i \in \mathbb{R}^{d'+1}$ using:

$$x_1 = \cos(\theta_1), \quad x_2 = \sin(\theta_1)\cos(\theta_2), \ldots, \quad x_{d'+1} = \sin(\theta_1)\sin(\theta_2)\cdots\sin(\theta_{d'}).$$

    Step 2.3: Embed Sample $\mathbf{x}_i$ into Ambient Space
    Map the sample $\mathbf{x}_i$ to the ambient space using:

$$\mathbf{y}_i = \mathbf{Q}\mathbf{x}_i$$

    Append $\mathbf{y}_i$ to the dataset $\mathbf{Y}$
end for
Return: The final dataset $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n]$

$$x_j = \sin\left(\mathbf{a}_j^\top \mathbf{z}\right) + \varepsilon_j,$$

where $\varepsilon_j \sim \mathcal{N}(0, \sigma_a^2)$ is Gaussian noise with variance σ_a^2. In our experiments, we set $\sigma_a^2 = 10^{-3}$.

3. Constructing the Dataset Samples The final samples $\mathbf{y} \in \mathbb{R}^d$ are formed by concatenating the ambient coordinates $x_1, x_2, \ldots, x_{d-d'}$ with the latent variables $z_1, z_2, \ldots, z_{d'}$:

$$\mathbf{y} = \left[x_1, x_2, \ldots, x_{d-d'}, z_1, z_2, \ldots, z_{d'}\right]^\top.$$

E.3 Error Metrics for Evaluation of Pullback Geometries

Geodesic Error. The geodesic error measures the difference between geodesics on the learned and ground truth pullback manifolds. Given two points $\mathbf{x}_0, \mathbf{x}_1 \in \mathbb{R}^d$, let $\gamma^{\varphi_{\theta_2}}_{\mathbf{x}_0,\mathbf{x}_1}(t)$ and $\gamma^{\varphi_{\mathrm{GT}}}_{\mathbf{x}_0,\mathbf{x}_1}(t)$ denote the geodesics induced by the learned map φ_{θ_2} and the ground truth map φ_{GT}, respectively, where $t \in [0,1]$.

The geodesic error is calculated as the mean Euclidean distance between the learned and ground truth geodesics over N pairs of points:

$$\text{Geodesic Error} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{T}\sum_{k=1}^{T}\left\|\gamma^{\varphi_{\theta_2}}_{\mathbf{x}_0^{(i)},\mathbf{x}_1^{(i)}}(t_k) - \gamma^{\varphi_{\mathrm{GT}}}_{\mathbf{x}_0^{(i)},\mathbf{x}_1^{(i)}}(t_k)\right\|_2,$$

where T is the number of time steps used to discretize the geodesic, and $t_k = \frac{k-1}{T-1}$ for $k = 1, \ldots, T$.

This metric captures the average discrepancy between the learned and ground truth geodesics, reflecting the accuracy of the learned pullback manifold.

Variation Error. The variation error quantifies the sensitivity of the geodesic computation under small perturbations to one of the endpoints. For two points $\mathbf{x}_0, \mathbf{x}_1 \in \mathbb{R}^d$, let $\mathbf{z} = \mathbf{x}_1 + \Delta\mathbf{x}$, where $\Delta\mathbf{x}$ is a small perturbation applied to $\mathbf{x}_1$. Define $\gamma^{\varphi_{\theta_2}}_{\mathbf{x}_0,\mathbf{x}_1}(t)$ and $\gamma^{\varphi_{\theta_2}}_{\mathbf{x}_0,\mathbf{z}}(t)$ as the geodesics from $\mathbf{x}_0$ to $\mathbf{x}_1$ and $\mathbf{z}$, respectively, induced by the learned map φ_{θ_2}.

The variation error is calculated as the mean Euclidean distance between the geodesic from $\mathbf{x}_0$ to $\mathbf{x}_1$ and the perturbed geodesic from $\mathbf{x}_0$ to $\mathbf{z}$:

$$\text{Variation Error} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{T}\sum_{k=1}^{T}\left\|\gamma^{\varphi_{\theta_2}}_{\mathbf{x}_0^{(i)},\mathbf{x}_1^{(i)}}(t_k) - \gamma^{\varphi_{\theta_2}}_{\mathbf{x}_0^{(i)},\mathbf{z}^{(i)}}(t_k)\right\|_2,$$

Algorithm 4 Sinusoid(d', d) Dataset Generation

Input: Intrinsic dimension d', ambient dimension d, number of samples n, variance $\sigma_m^2 = 3$, noise variance $\sigma_a^2 = 10^{-3}$
Output: Dataset $\mathbf{Y} \in \mathbb{R}^{n \times d}$
Step 1: Generate Shear Vectors
for $j = 1$ to $d - d'$ **do**
    Sample shear vector $\mathbf{a}_j \in \mathbb{R}^{d'}$ from Uniform$(1,2)^{d'}$
end for
Step 2: Construct Dataset
for $i = 1$ to n **do**
    Step 2.1: Sample Latent Variables
    Generate latent variables $\mathbf{z}_i \in \mathbb{R}^{d'}$ from a multivariate Gaussian:

$$\mathbf{z}_i \sim \mathcal{N}(0, \sigma_m^2 \cdot I_{d'})$$

    Step 2.2: Compute Ambient Coordinates for Sample i
    for $j = 1$ to $d - d'$ **do**
        Compute ambient coordinate x_j for the i-th sample:

$$x_j = \sin\left(\mathbf{a}_j^\top \mathbf{z}_i\right) + \varepsilon_j, \quad \varepsilon_j \sim \mathcal{N}(0, \sigma_a^2)$$

    end for
    Step 2.3: Form Final Sample $\mathbf{y}_i$
    Concatenate the ambient coordinates $\mathbf{x} = [x_1, x_2, \ldots, x_{d-d'}]$ and the latent variables $\mathbf{z}_i$ to form the final sample $\mathbf{y}_i \in \mathbb{R}^d$:

$$\mathbf{y}_i = [x_1, x_2, \ldots, x_{d-d'}, z_1, z_2, \ldots, z_{d'}]^\top$$

    Append $\mathbf{y}_i$ to the dataset $\mathbf{Y}$
end for
Return: The final dataset $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n]$

where N is the number of sampled point pairs, T is the number of time steps used to discretize the geodesic, and $t_k = \frac{k-1}{T-1}$ for $k = 1, \ldots, T$.

This metric evaluates the robustness of the learned geodesic against small perturbations, providing insight into the stability of the learned manifold.

E.4 Training Details

The following section describes the important configuration parameters for reproducing the experiments on manifold mappings. All experiments share some common parameters, which are listed below, while dataset-specific parameters are provided in Table E.1.

Common Parameters:

- **Optimizer:** Adam with `betas` = (0.9, 0.99), `eps` = 1×10^{-8}, and weight decay of 1×10^{-5}.
- **Learning Rate Schedule:** Warm-up cosine annealing with 1000 warm-up steps.
- **Gradient Clipping:** Gradient norm clipped to 1.0.
- **Model Architecture:** A composition of affine coupling layers is used, where each layer transforms part of the input while keeping the other part unchanged. The transformation function in each layer is modeled by a residual network (ResNet) consisting of 64 hidden features, 2 residual blocks, ReLU activations, and no batch normalization. Dropout is set to 0, and transformations alternate across different dimensions at each layer.

Table E.1 Training configurations for each experiment.

Dataset	Flow Steps	Epochs	Batch Size	λ_{iso}	λ_{vol}	Learning Rate
Sinusoid(1,3)	8	1000	64	1.0	1.0	3×10^{-4}
Sinusoid(2,3)	8	1000	64	1.0	1.0	3×10^{-4}
Sinusoid(5,20)	24	2000	128	1.2	2.5	4×10^{-4}
Hemisphere(2,3)	8	2000	64	1.0	1.0	4×10^{-4}
Hemisphere(5,20)	12	2000	64	0.75	1.2	4×10^{-4}

E.5 Data Manifold Approximation

The learned manifold, shown in orange in 7.1, is the set $D_\varepsilon(\mathscr{U})$, where D_ε is the RAE decoder 7.21, the set $\mathscr{U}$ in the latent space is the open set given by

$$\mathscr{U} = \prod_{i=1}^{d_\varepsilon} (-3\sqrt{\mathbf{a}_{u_i}}, 3\sqrt{\mathbf{a}_{u_i}})$$

and $\mathbf{a}_{u_1}, \ldots, \mathbf{a}_{u_{d_\varepsilon}}$ are the d_ε highest learned variances corresponding to the ones used in the RAE construction.

To visualize this in practice, we construct a mesh grid by linearly sampling each latent dimension from $-3\sqrt{\mathbf{a}_{u_i}}$ to $+3\sqrt{\mathbf{a}_{u_i}}$, for $i = 1, \ldots, d_\varepsilon$, where d_ε is the number of significant latent dimensions. Practically, the off-manifold latent dimensions (those corresponding to negligible variances) are set to zero. The decoder D_ε then maps this grid from $\mathscr{U}$ back to $\mathbb{R}^d$, generating an approximation of the data manifold, as illustrated in 7.1.