Handbook of Generative Models for Mathematicians

Xuda Ye

Purdue University

December 13, 2025

Generative models have become a cornerstone of modern machine learning. While many computer scientists and engineers are actively researching this field, their notations and proofs, often stemming from different conventions and mathematical training, can be ambiguous to mathematicians and statisticians. Common examples include writing expectations without explicitly stating the underlying distribution or using the same notation (e.g., x_t) to represent both a random variable and its realization. Furthermore, abbreviations are frequently employed without their corresponding full terms being defined, adding to the ambiguity.

This handbook aims to bridge this gap by providing mathematically rigorous (at least formally) formulations of generative models. It also elucidates the theoretical justifications for why minimizing their associated loss functions leads to high-quality sample generation.

In this setup, we consider a target distribution $\pi(x)$ on $\mathbb{R}^d$, which is typically accessible only through a set of given samples $\{X_i\}_{i=1}^N$ drawn from it. The goal of a generative model is to produce new samples that follow $\pi(x)$, or equivalently, samples that are statistically indistinguishable from the provided data $\{X_i\}_{i=1}^N$.

A common strategy to achieve this is to define a simple, tractable base distribution $p_0(x)$ on $\mathbb{R}^d$ (e.g., a standard Gaussian) and then find an underlying dynamical process that gradually transports $p_0(x)$ to a terminal distribution $p_T(x)$ that approximates the target, $p_T(x) \approx \pi(x)$. This process is characterized by a flow of distributions $\{p_t(x)\}_{0 \leqslant t \leqslant T}$. The underlying dynamics can be either deterministic or stochastic. The principal methods include:

- Normalizing Flow [1, 2, 3]
- Flow Matching [4]
- Score-Based Diffusion [5]

This handbook does not include Denoising Diffusion Probabilistic Models (DDPM) [6]. While DDPM remains a foundational benchmark, its methodology is largely encompassed and generalized by the Score-Based Diffusion framework, which is covered in detail. Furthermore, the exposition in the original DDPM paper presents notational and formal ambiguities that are difficult to reconcile with the standards of mathematical rigor this text aims to uphold.

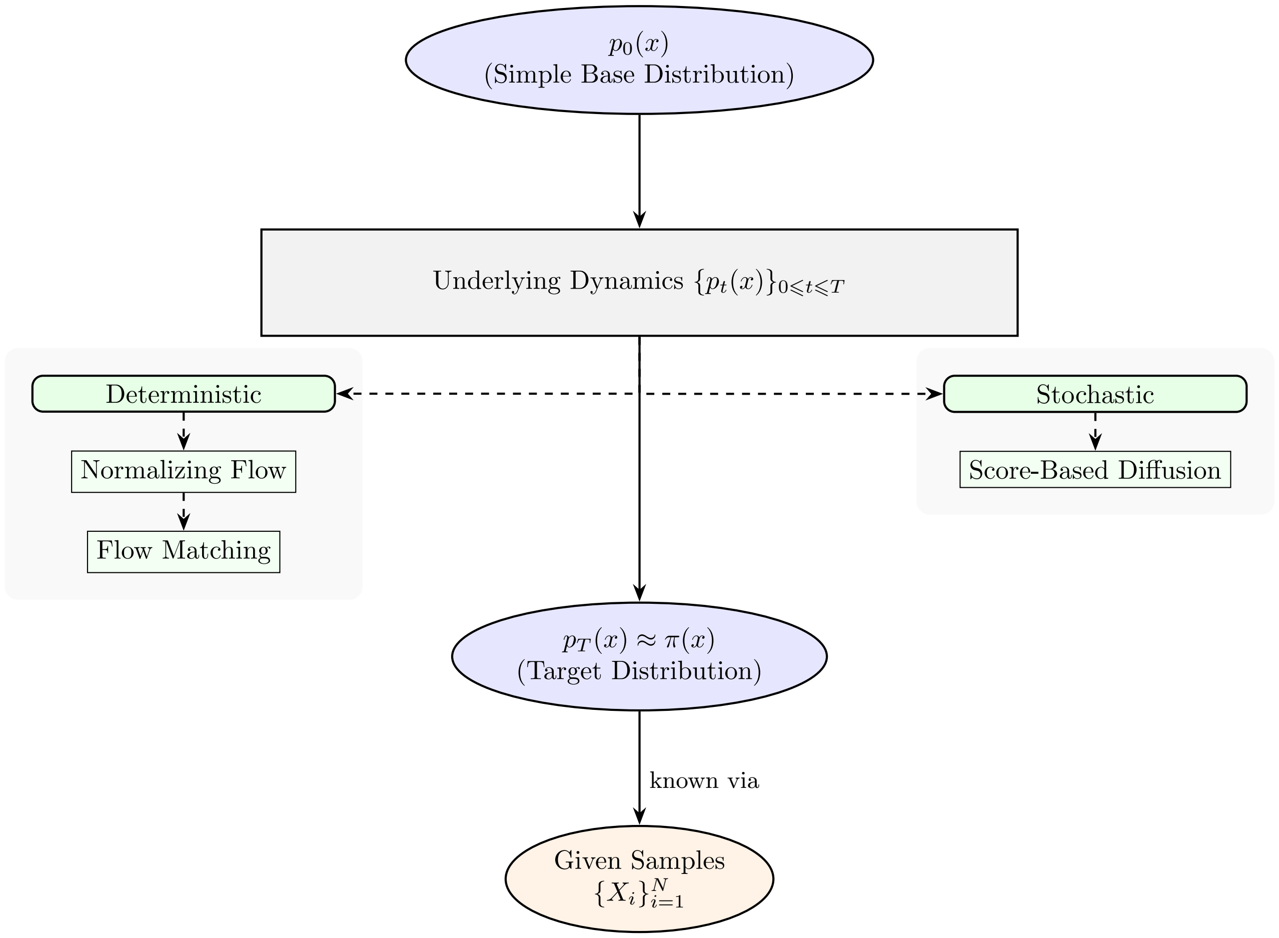

Figure 1: A flowchart of modern generative models.

1 Normalizing Flow

We aim to find an deterministic and invertible map $f : \mathbb{R}^d \to \mathbb{R}^d$ such that f transports the base distribution p_0 to approximate the target π, namely,

$$f_{\#}p_0 \approx \pi \quad \Longleftrightarrow \quad p_0 \approx f_{\#}^{-1}\pi.$$

For notational convenience, we let z and x be the variables for $p_0(z)$ and $\pi(x)$, respectively. The change of variables formula implies the density of $f_{\#}p_0$ is

$$(f_{\#}p_0)(x) = \frac{p_0(z)}{|\det J_z f(z)|}, \qquad x = f(z). \tag{1}$$

Here, $J_z f(z) \in \mathbb{R}^{d\times d}$ denotes the Jacobian matrix of the map $f(z)$.

Next, a natural idea is to minimize the KL divergence between $f_{\#}p_0(x)$ and $\pi(x)$:

$$\begin{aligned} D_{\mathrm{KL}}(\pi \| f_{\#}p_0) &= \int_{\mathbb{R}^d} \pi(x) \log \frac{\pi(x)}{(f_{\#}p_0)(x)} \mathrm{d}x \\ &= \mathrm{const} - \int_{\mathbb{R}^d} \pi(x) \log(f_{\#}p_0)(x)\mathrm{d}x \\ &= \mathrm{const} - \int_{\mathbb{R}^d} \pi(x)\Big(\log p_0\big(f^{-1}(x)\big) - \log\big|\det J_z f\big(f^{-1}(x)\big)\big|\Big)\mathrm{d}x, \end{aligned}$$

where we have applied the logarithm to (1) and substituted $z = f^{-1}(x)$. Therefore, minimizing this KL divergence is equivalent to maximizing the functional

$$\boxed{-\mathcal{L}[f] = \int_{\mathbb{R}^d} \pi(x)\Big(\log p_0\big(f^{-1}(x)\big) - \log\big|\det J_z f\big(f^{-1}(x)\big)\big|\Big)\mathrm{d}x.} \tag{2}$$

In practice, the expectation with respect to $\pi(x)$ is approximated by an empirical average over the samples $\{X_i\}_{i=1}^N$, and f is parameterized by a neural network. The quantity in (2) is also called the Evidence Lower Bound (ELBO), although I do not favor this name.

We now introduce how to parameterize this map f using a neural network. A crucial requirement for this parameterization is that the Jacobian matrix $J_z f(z)$ must be explicitly computable with respect to the network parameters.

1.1 Realization in coupling flow

A simple approach to parameterize f is to write f as the composition

$$f = f_K \circ \cdots \circ f_1, \tag{3}$$

where each $f_k : \mathbb{R}^d \to \mathbb{R}^d$ is an invertible map designed such that the determinant of its Jacobian, Jf_k, is efficiently computable. Letting $z_0 = z$ and $z_k = f_k(z_{k-1})$ for $k = 1, \dots, K$, the chain rule for determinants implies

$$\log|\det J_z f(z)| = \sum_{k=1}^{K} \log|\det J_{z_{k-1}} f_k(z_{k-1})|. \tag{4}$$

A simple and widely-used choice for f_k is the **coupling flow**.

The core idea of a coupling flow is to partition the input vector and transform one part using parameters generated by the other part. This construction yields a block-triangular Jacobian matrix, for which the determinant is trivial to compute.

Formally, for each layer f_k, we partition its input $z_{k-1} \in \mathbb{R}^d$ into two disjoint parts. Let $d_A < d$, $d_B = d - d_A$, and let the partition be $z_{k-1} = (z_A, z_B)$, where $z_A \in \mathbb{R}^{d_A}$ and $z_B \in \mathbb{R}^{d_B}$. The coupling flow f_k computes its output $z_k = (x_A, x_B)$ by leaving one part unchanged (an identity) and applying a simple invertible transformation to the other.

Specifically, an **affine coupling layer** is defined as:

$$\begin{aligned} x_A &= z_A \\ x_B &= z_B \odot \exp(\mathcal{S}(z_A)) + \mathcal{T}(z_A) \end{aligned}$$

where $\odot$ denotes element-wise multiplication, and $\mathcal{S}, \mathcal{T} : \mathbb{R}^{d_A} \to \mathbb{R}^{d_B}$ are neural networks (e.g., MLPs) parameterized by θ_k. The exponential function is used to ensure the scaling factors $\exp(\mathcal{S}(z_A))$ are strictly positive, guaranteeing invertibility.

The critical advantage of this construction is its Jacobian determinant. The Jacobian matrix of f_k with respect to z_{k-1} has a block-lower-triangular structure:

$$J_{z_{k-1}} f_k(z_{k-1}) = \begin{pmatrix} \dfrac{\partial x_A}{\partial z_A} & \dfrac{\partial x_A}{\partial z_B} \\ \dfrac{\partial x_B}{\partial z_A} & \dfrac{\partial x_B}{\partial z_B} \end{pmatrix} = \begin{pmatrix} I & 0 \\ \dfrac{\partial x_B}{\partial z_A} & \operatorname{diag}(\exp(\mathcal{S}(z_A))) \end{pmatrix}$$

The determinant of a triangular matrix is the product of its diagonal entries. Therefore, the log-determinant is simply the sum of the outputs of the scaling network $\mathcal{S}$:

$$\log |\det J_{z_{k-1}} f_k(z_{k-1})| = \log \left(\prod_{i=1}^{d_B} \exp(\mathcal{S}(z_A))_i \right) = \sum_{i=1}^{d_B} \mathcal{S}(z_A)_i. \tag{5}$$

This value is computationally efficient, as it is a direct output of the network. To ensure the entire map f is expressive and that all variables can be transformed, the partition is typically alternated or permuted between subsequent layers (e.g., f_{k+1} transforms the part that f_k left as an identity).

In practice, the coupling flow requires an input dimension $d \geqslant 2$. To ensure the entire map is expressive and that all variables are eventually transformed, the partition (z_A, z_B) must be alternated or permuted between subsequent layers. For example, common strategies to define z_A and z_B include using a "checkerboard" pattern or splitting by "channel".

1.2 Continuous dynamics parameterization

Although the coupling flow construction is straightforward, its expressive power is often limited by the simple functional form of the coupling layers. A more popular and flexible approach, known as a **continuous normalizing flow**, is to define f as the flow map of an ODE.

This involves parameterizing a time-dependent velocity field $u(x, t)$ on $\mathbb{R}^d \times [0, T]$. The final map f is then defined as the time-T map $z \mapsto x(T; z)$, where $\{x(t; z)\}_{0 \leqslant t \leqslant T}$ is the solution to the initial value problem:

$$\begin{cases} \dfrac{\mathrm{d}}{\mathrm{d}t} x(t; z) = u(x(t; z), t), \\ x(0; z) = z. \end{cases} \tag{6}$$

A crucial property of this ODE-based parameterization is that the *logarithm* of the Jacobian determinant can be computed efficiently. Using the instantaneous change of volume formula (a consequence of Jacobi's formula), we have

$$\log|\det J_z f(z)| = \int_0^T \mathrm{tr}\Big(J_x u(x(t;z),t)\Big)\mathrm{d}t. \tag{7}$$

As a consequence, the required log-determinant can be conveniently computed by integrating the trace of the velocity field's Jacobian, $\mathrm{tr}(J_x u)$, along the trajectory of the ODE (6). This is typically solved numerically by augmenting the original ODE system.

A proof of the equality (7) can be established as follows. Taking the gradient with respect to z in the ODE (6), we obtain the sensitivity equation satisfied by the Jacobian matrix $J_z x(t;z) \in \mathbb{R}^{d\times d}$:

$$\begin{cases} \dfrac{\mathrm{d}}{\mathrm{d}t} J_z x(t;z) = J_x u(x(t;z),t) J_z x(t;z), \\ J_z x(0;z) = I_d. \end{cases}$$

By Jacobi's formula, which gives the derivative of the determinant, we obtain

$$\frac{\mathrm{d}}{\mathrm{d}t}\log|\det J_z x(t;z)| = \mathrm{tr}\Big(J_x u(x(t;z),t)\Big).$$

Integrating this expression over the time interval $[0,T]$ and using the initial condition $\log|\det J_z x(0;z)| = \log|\det I_d| = 0$ produces the desired result (7).

Substituting the log-determinant formula (7) into the loss function (2), we obtain the objective for the continuous normalizing flow, expressed in terms of the velocity field $u(x,t)$:

$$\boxed{-\mathcal{L}[u] = \int_{\mathbb{R}^d} \pi(x)\Bigg(\log p_0\big(f^{-1}(x)\big) - \int_0^T \mathrm{tr}\Big(J_x u\big(x(t;f^{-1}(x)),t\big)\Big)\mathrm{d}t\Bigg)\mathrm{d}x.} \tag{8}$$

While this loss functional (8) appears complex, the procedure to compute its stochastic approximation (i.e., the Monte Carlo estimate using the dataset $\{X_i\}_{i=1}^N$) is straightforward:

1. Sample x_* from the data distribution $\pi(x)$ (or the dataset $\{X_i\}_{i=1}^N$).
2. Compute $z_* = f^{-1}(x_*)$ by solving the ODE (6) backward in time from $t = T$ to $t = 0$, using $x_* = x(T;z_*)$ as the terminal condition.
3. Compute the integrand of (8) (the "sample loss") for this x_*:

   $$\log p_0(z_*) - \int_0^T \mathrm{tr}\Big(J_x u(x(t;z_*),t)\Big)\mathrm{d}t,$$

   where the integral is computed by solving the ODE (6) forward in time, starting from z_*, to obtain the full trajectory $\{x(t;z_*)\}_{0\leqslant t\leqslant T}$.

For numerical implementation, the ODEs in steps 2 and 3 must be solved using a numerical time-discretization scheme.

2 Flow Matching

Flow matching is arguably one of the most popular generative models at present, primarily because it combines a relatively simple formulation with straightforward training. Let $p_0(x)$ and $p_1(x)$ be the base and target distributions on $\mathbb{R}^d$, respectively. The objective is to construct a time-dependent velocity field $u(x,t)$ on $\mathbb{R}^d \times [0,1]$ such that the flow generated by the ODE

$$\frac{\mathrm{d}x}{\mathrm{d}t} = u(x,t)$$

transports $p_0(x)$ to $p_1(x)$.

Let x_0 and x_1 be independent random variables sampled from $p_0(x)$ and $p_1(x)$, respectively. We then consider the following simple loss function:

$$\boxed{\mathcal{L}[u] = \int_0^1 \mathbb{E}_{x_0,x_1}\Big[\big|u\big((1-t)x_0 + tx_1, t\big) - (x_1 - x_0)\big|^2\Big]\mathrm{d}t.} \tag{9}$$

Here, the loss minimizes the expected squared difference between the parameterized velocity field $u(x,t)$ and the constant velocity vector $(x_1 - x_0)$ along the linear path $x_t = (1-t)x_0 + tx_1$.

We will now demonstrate that the minimizer of $\mathcal{L}[u]$ in (9) indeed induces the flow that transports $p_0(x)$ to $p_1(x)$. On one hand, the optimal velocity field $u(x,t)$ that minimizes (9) is given by the conditional expectation:

$$u(x,t) = \mathbb{E}_{x_0,x_1}\Big[x_1 - x_0 \;\Big|\; tx_1 + (1-t)x_0 = x\Big]. \tag{10}$$

This means $u(x,t)$ is the conditional expectation of $x_1 - x_0$ given that $tx_1 + (1-t)x_0 = x$. On the other hand, we define the probability flow $\{p_t(x)\}_{0\leqslant t\leqslant 1}$ by

$$p_t(x) = \mathbb{E}_{x_0,x_1}\Big[\delta\big(x - tx_1 - (1-t)x_0\big)\Big], \tag{11}$$

namely, $p_t(x)$ is the marginal distribution of the interpolation point $tx_1 + (1-t)x_0$ at time t. As a consequence, $u(x,t)$ can be equivalently written as

$$u(x,t) = \frac{1}{p_t(x)}\mathbb{E}_{x_0,x_1}\Big[(x_1 - x_0)\delta\big(x - tx_1 - (1-t)x_0\big)\Big].$$

In what follows, we show that $u(x,t)$ exactly generates the probability flow $p_t(x)$, i.e., it satisfies the continuity equation:

$$\frac{\partial p_t(x)}{\partial t} + \nabla \cdot (p_t(x)u(x,t)) = 0. \tag{12}$$

We prove (12) in the sense of distributions. By choosing a test function $\phi(x)$ with compact support in $\mathbb{R}^d$, equation (12) is equivalent to its weak form:

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\mathbb{R}^d} p_t(x)\phi(x)\mathrm{d}x = \int_{\mathbb{R}^d} p_t(x)u(x,t)\cdot\nabla\phi(x)\mathrm{d}x. \tag{13}$$

We compute the LHS and RHS of (13) separately. For the LHS of (13), we have:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\int_{\mathbb{R}^d} p_t(x)\phi(x)\mathrm{d}x &= \mathbb{E}_{x_0,x_1}\left[\frac{\mathrm{d}}{\mathrm{d}t}\int_{\mathbb{R}^d}\delta\big(x - tx_1 - (1-t)x_0\big)\phi(x)\mathrm{d}x\right]\\ &= \mathbb{E}_{x_0,x_1}\left[\frac{\mathrm{d}}{\mathrm{d}t}\phi\big(tx_1 + (1-t)x_0\big)\right]\\ &= \mathbb{E}_{x_0,x_1}\Big[(x_1 - x_0)\cdot\nabla\phi\big(tx_1 + (1-t)x_0\big)\Big].\end{aligned}$$

For the RHS of (13), we use $p_t(x)u(x,t) = \mathbb{E}_{x_0,x_1}[(x_1 - x_0)\delta(x - tx_1 - (1-t)x_0)]$:

$$\begin{aligned}\int_{\mathbb{R}^d} p_t(x)u(x,t)\cdot\nabla\phi(x)\mathrm{d}x &= \int_{\mathbb{R}^d}\mathbb{E}_{x_0,x_1}\Big[(x_1 - x_0)\delta\big(x - tx_1 - (1-t)x_0\big)\Big]\cdot\nabla\phi(x)\mathrm{d}x\\ &= \mathbb{E}_{x_0,x_1}\left[(x_1 - x_0)\cdot\int_{\mathbb{R}^d}\delta\big(x - tx_1 - (1-t)x_0\big)\nabla\phi(x)\mathrm{d}x\right]\\ &= \mathbb{E}_{x_0,x_1}\Big[(x_1 - x_0)\cdot\nabla\phi\big(tx_1 + (1-t)x_0\big)\Big].\end{aligned}$$

Since the LHS and RHS of (13) are equal, the continuity equation (12) is satisfied. This implies that the optimal velocity field $u(x,t)$ learned from the loss function (9) correctly generates the probability flow $p_t(x)$ that transports $p_0(x)$ to $p_1(x)$.

In principle, the random variables x_0 and x_1 do not need to be sampled independently from $p_0(x)$ and $p_1(x)$. In fact, let $\gamma \in \Pi(p_0, p_1)$ be an arbitrary coupling (joint distribution) between $p_0(x)$ and $p_1(x)$. We can find a velocity field u by minimizing the more general loss functional:

$$\mathcal{L}[u] = \int_0^1 \mathbb{E}_\gamma\Big[\big|u\big((1-t)x_0 + tx_1, t\big) - (x_1 - x_0)\big|^2\Big]\mathrm{d}t,$$

The optimal velocity field u from this minimization generates a flow map ϕ_t. This flow, in turn, induces a new coupling γ' (e.g., $\gamma' = (\mathrm{Id}, \phi_1)_{\#}p_0$). This framework, known as Rectified Flow, defines an operator that maps one coupling γ to another γ'. It has been shown that iterating this procedure can lead to quadratic convergence to the optimal transport map [7].

3 Score-Based Diffusion

Score-based diffusion models are a foundational class of generative models. The methodology consists of two complementary processes. First, a **forward process** $\{q_t(x)\}_{0\leqslant t\leqslant T}$ is defined to gradually transform the target distribution $q_0(x) = \pi(x)$ into a tractable base distribution $q_T(x)$. Second, a **reverse process** $\{p_t(x)\}_{0\leqslant t\leqslant T}$ is constructed to approximate the time-reversal of the forward process. This reverse process, which starts from $q_T(x)$ and aims to recover $q_0(x)$, is characterized by the **score function** $\nabla \log q_t(x)$ of the forward process. The core task is to learn this score function, typically via a neural network. Once learned, simulating the reverse process allows for the generation of new samples approximating the target distribution.

Specifically, the forward diffusion process is defined by the stochastic differential equation (SDE):

$$\mathrm{d}y_t = f(y_t, t)\mathrm{d}t + g(t)\mathrm{d}B_t, \tag{14}$$

where B_t is the standard Brownian motion in $\mathbb{R}^d$. This SDE induces a flow of distributions $\{q_t(x)\}_{0\leqslant t\leqslant T}$, which evolves according to the corresponding Fokker–Planck equation:

$$\frac{\partial q_t(x)}{\partial t} = -\nabla\cdot\big(f(x,t)q_t(x)\big) + \frac{1}{2}g^2(t)\Delta q_t(x), \quad t\in[0,T]. \tag{15}$$

This process is designed to transform the initial data distribution $q_0(x) = \pi(x)$ into a simple, tractable base distribution $q_T(x)$.

The objective is to construct a reverse process, whose probability flow $\{p_t(x)\}_{0\leqslant t\leqslant T}$ satisfies the time-reversibility condition $p_t(x) = q_{T-t}(x)$. By replacing $t\to T-t$ and substituting $p_t(x) = q_{T-t}(x)$ into the Fokker–Planck equation, we find that $p_t(x)$ must satisfy the following PDE:

$$\frac{\partial p_t(x)}{\partial t} = \nabla\cdot\big(f(x,T-t)p_t(x)\big) - \frac{1}{2}g^2(T-t)\Delta p_t(x), \quad t\in[0,T]. \tag{16}$$

This equation, however, is ill-posed. The negative sign on the diffusion term, $-\frac{1}{2}g^2(T-t)$, makes it an instance of the backward heat equation, which is notoriously unstable to solve numerically. In general, it is impossible to directly reverse a diffusion process via its Fokker–Planck equation without additional information.

A stable reverse process can be constructed by incorporating the score function. Using the identity $p_t(x) = q_{T-t}(x)$, we have the equality:

$$\nabla\cdot\big(g^2(T-t)\nabla\log q_{T-t}(x)p_t(x)\big) = \nabla\cdot\big(g^2(T-t)\nabla q_{T-t}(x)\big) = g^2(T-t)\Delta p_t(x).$$

This identity allows us to rewrite the ill-posed equation (16) by manipulating its diffusion term. An algebraic rearrangement (effectively adding and subtracting terms related to the score) yields

$$\frac{\partial p_t(x)}{\partial t} = \nabla\cdot\Big(\big(f(x,T-t) - g^2(T-t)\nabla\log q_{T-t}(x)\big)p_t(x)\Big) + \frac{1}{2}g^2(T-t)\Delta p_t(x). \tag{17}$$

This equation is now a well-posed forward Fokker–Planck equation. Therefore, with explicit knowledge of the score function $\nabla\log q_t(x)$ for $t\in[0,T]$, one can simulate the reverse process by solving the corresponding SDE:

$$\boxed{\mathrm{d}x_t = \big(-f(x_t,T-t) + g^2(T-t)\nabla\log q_{T-t}(x_t)\big)\mathrm{d}t + g(T-t)\mathrm{d}\bar{B}_t, \quad t\in[0,T].} \tag{18}$$

where $\bar{B}_t$ is a standard Brownian motion. Alternatively, by rearranging the terms, the ill-posed equation (16) can be expressed in the equivalent form of a pure continuity equation:

$$\frac{\partial p_t(x)}{\partial t} = \nabla\cdot\Bigg(\Big(f(x,T-t) - \frac{1}{2}g^2(T-t)\nabla\log q_{T-t}(x)\Big)p_t(x)\Bigg). \tag{19}$$

This formulation describes a deterministic probability flow. Therefore, the reverse process can also be constructed by solving the corresponding ODE, which defines the characteristic curves:

$$\boxed{\mathrm{d}x_t = \Big(-f(x_t,T-t) + \frac{1}{2}g^2(T-t)\nabla\log q_{T-t}(x_t)\Big)\mathrm{d}t, \quad t\in[0,T].} \tag{20}$$

In summary, the central task is to learn the score function $\{\nabla\log q_t(x)\}_{0\leqslant t\leqslant T}$. Once this function is estimated (typically via a neural network), one can simulate either the reverse SDE (18) or the reverse ODE (20). This procedure, initiated by sampling from the base distribution $q_T(x)$, transforms these samples into new samples that approximate the desired data distribution $\pi(x)$.

We now describe how the score function $s(x,t) = \nabla \log q_t(x)$ is learned. For notational clarity, let $Q_{t|0}(\cdot|y_0)$ denote the conditional probability distribution of y_t given y_0, which we distinguish from the marginal probability flow $q_t(x)$. The score function $s(x,t)$ is trained by minimizing the following L^2-loss functional, commonly known as a score-matching objective:

$$\boxed{\mathcal{L}[s] = \int_0^T \mathbb{E}_{y_0\sim\pi, y_t\sim Q_{t|0}(\cdot|y_0)}\left[\left|s(y_t,t) - \nabla \log Q_{t|0}(y_t|y_0)\right|^2\right]\lambda(t)\mathrm{d}t,} \tag{21}$$

where $\lambda(t)$ is a positive weighting function. The expectation is taken over the initial data $y_0 \sim q_0(\cdot) = \pi(\cdot)$ and the corresponding state y_t at time t generated from y_0 by the forward SDE (14). This objective trains the network $s(x,t)$ to match the true score of the conditional transition kernel.

Finally, we must verify that the minimizer of (21) coincides with the true score function, $\nabla \log q_t(x)$. In fact, the minimizer $s(x,t)$ of the L^2-loss (21) is the conditional expectation of the target given the input, which is readily given by

$$s(x,t) = \mathbb{E}_{y_0}\left[\nabla \log Q_{t|0}(x|y_0)\,|\,y_t = x\right] = \int_{\mathbb{R}^d} \nabla_x \log Q_{t|0}(x|y_0)\, Q_{0|t}(y_0|x)\,\mathrm{d}y_0. \tag{22}$$

Here, $Q_{0|t}(y_0|x)$ is the posterior distribution of the initial state y_0 given the observation $y_t = x$. According to Bayes' theorem,

$$Q_{0|t}(y_0|x) = \frac{Q_{t|0}(x|y_0)\, q_0(y_0)}{q_t(x)} = \frac{Q_{t|0}(x|y_0)\, \pi(y_0)}{q_t(x)}. \tag{23}$$

Therefore, substituting this into (22), the optimal $s(x,t)$ of $\mathcal{L}[s]$ is

$$\begin{aligned}
s(x,t) &= \int_{\mathbb{R}^d} \left(\frac{\nabla_x Q_{t|0}(x|y_0)}{Q_{t|0}(x|y_0)}\right)\left(\frac{Q_{t|0}(x|y_0)\pi(y_0)}{q_t(x)}\right)\mathrm{d}y_0 \\
&= \frac{1}{q_t(x)}\int_{\mathbb{R}^d} \pi(y_0)\nabla_x Q_{t|0}(x|y_0)\mathrm{d}y_0 \\
&= \frac{1}{q_t(x)}\nabla_x\left(\int_{\mathbb{R}^d} \pi(y_0)Q_{t|0}(x|y_0)\mathrm{d}y_0\right) \\
&= \frac{1}{q_t(x)}\nabla_x q_t(x) = \nabla_x \log q_t(x),
\end{aligned}$$

where we have used the definition of the marginal distribution $q_t(x) = \int \pi(y_0)Q_{t|0}(x|y_0)\mathrm{d}y_0$. This confirms that the minimizer of (21) is indeed the true score function.

4 Numerical Tests

In this section, we evaluate the performance of Normalizing Flow, Flow Matching, and Score-Based Diffusion on three simple 2D distributions: a Gaussian mixture, an I-beam, and an annulus. The target distributions are visualized in Figure 2. The sample trajectories, illustrating the evolution from a standard Gaussian distribution to each target, are presented for Normalizing Flow in Figure 3, Flow Matching in Figure 4, and Score-Based Diffusion in Figure 5.

The source codes for this project can be found at the following link: https://xuda-ye.wordpress.com/wp-content/uploads/2025/11/handbook-of-generative-models.zip

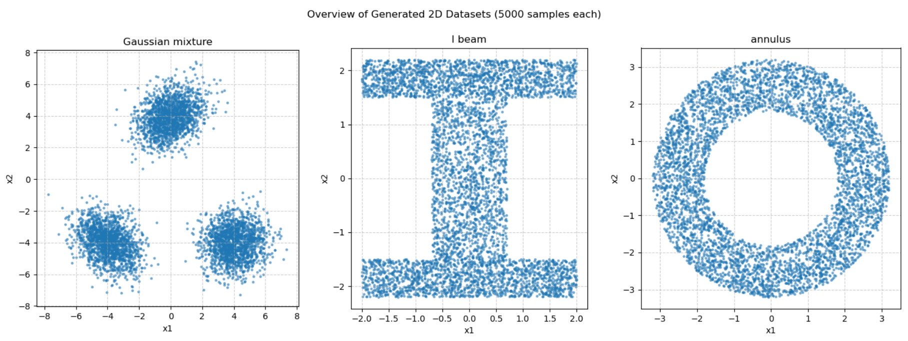

Figure 2: Target distributions: Gaussian mixture, I-beam and annulus.

References

[1] Ivan Kobyzev, Simon JD Prince, and Marcus A Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE transactions on pattern analysis and machine intelligence*, 43(11):3964–3979, 2020.

[2] Emile Mathieu and Maximilian Nickel. Riemannian continuous normalizing flows. *Advances in neural information processing systems*, 33:2503–2515, 2020.

[3] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021.

[4] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[5] Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. Maximum likelihood training of score-based diffusion models. *Advances in neural information processing systems*, 34:1415–1428, 2021.

[6] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[7] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

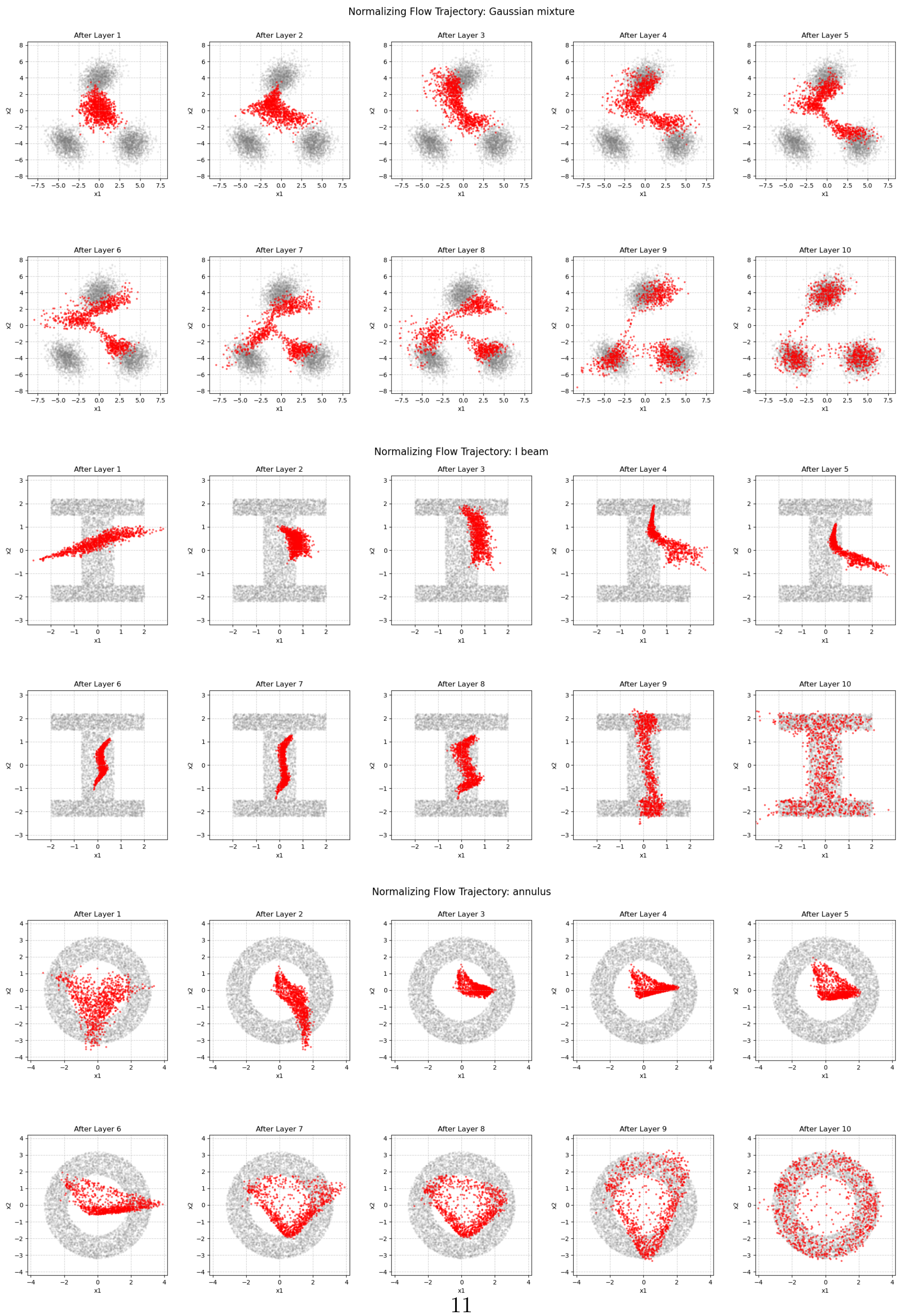

Figure 3: Sample trajectories of Normalizing Flow.

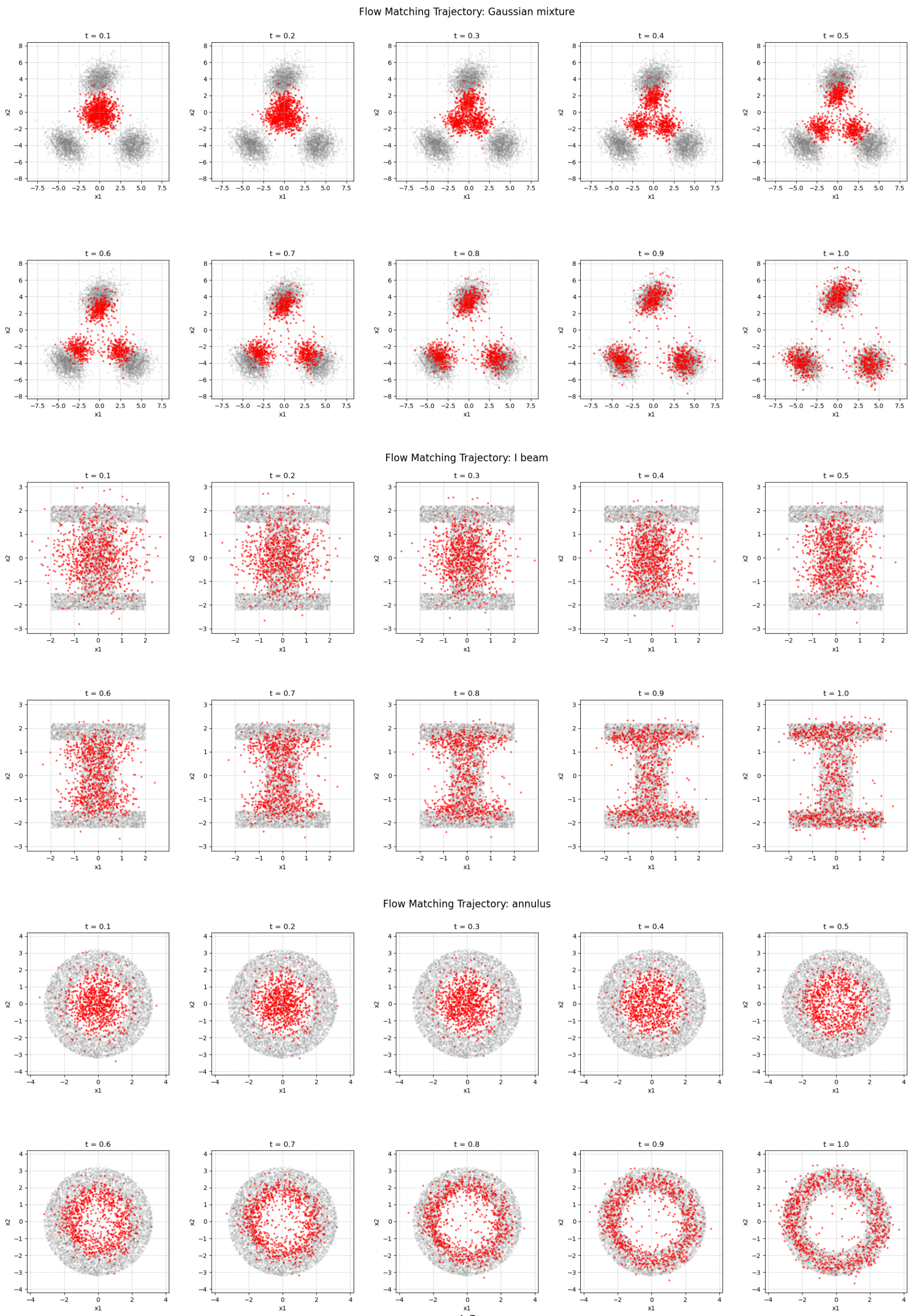

Figure 4: Sample trajectories of Flow Matching.

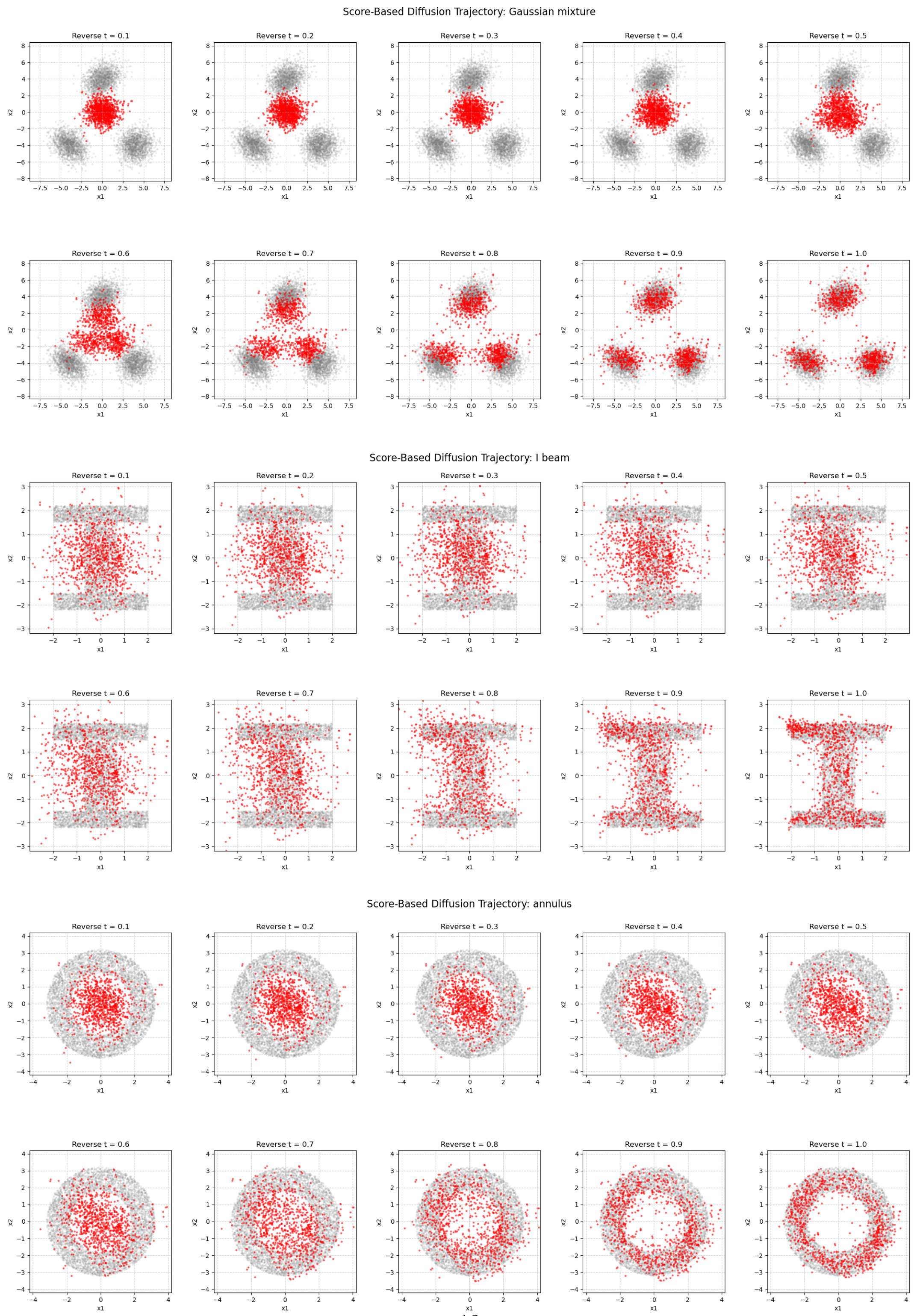

Figure 5: Sample trajectories of Score-Based Diffusion.