Generative Models: Foundations and Frontiers

Shuiwang Ji

sji@tamu.edu

https://people.tamu.edu/~sji/

Department of Computer Science & Engineering
Texas A&M University



Purpose and audience

1. **Purpose:** Develop a deep understanding of the mathematical foundations, principles, and connections of modern generative models, rather than merely learning how to use them.
2. **Audience:** Technical readers, including graduate students and researchers, who wish to understand how and why generative models work.
3. **Format:** These materials are essentially lecture notes presented in slide format. They include detailed mathematical explanations, derivations, and proofs rather than only high-level summaries.
4. **Completeness and self-containment:** The slides aim to be comprehensive and self-contained, providing the necessary background, reviews, and proofs for independent study. Completeness is prioritized over conciseness, so some repetition is intentional and unavoidable. Review materials and proofs can be skipped without disrupting the logical flow of the main presentation.
5. **Prerequisites:** A working knowledge of calculus, probability, and linear algebra is assumed.

Use of AI in preparing these materials

1. **Intellectual contributions:** The core ideas, conceptual perspectives, overall organization, and pedagogical approach originate from the author.
2. **Role of AI:** AI tools were used to accelerate and assist with selected aspects of material preparation, including:
   - Developing background explanations and supplementary mathematical details.
   - Expanding derivations and proofs.
   - Checking mathematical notation, logical consistency, and presentation clarity.
   - Refining language and slide formatting.
3. **Human oversight:** The author reviewed AI-assisted content, and retained responsibility for the final presentation.
4. **Guiding principle:** AI was used to improve efficiency and completeness, not to replace independent scientific reasoning or intellectual judgment.

Outline

1 Introduction
2 Flow Matching
3 Diffusion Models
4 Score Matching
5 Connections
6 Frontiers
7 Latent Variable Models
8 Variational Inference

What is generative modeling?

1. We observe a dataset of samples from an unknown data distribution:

$$\mathcal{D} = \{\boldsymbol{x}^{(n)}\}_{n=1}^{N}, \qquad \boldsymbol{x}^{(n)} \sim p_{\text{data}}(\boldsymbol{x}).$$

2. A generative model specifies a distribution $p_\theta(\boldsymbol{x})$ that approximates $p_{\text{data}}(\boldsymbol{x})$.
3. Its defining capability is to produce new samples with statistical properties resembling the training data:

$$\widetilde{\boldsymbol{x}} \sim p_\theta(\boldsymbol{x}).$$

4. Applications include images, text, audio, molecules, and physical states.

What does a generative model learn?

1. High-dimensional observations exhibit strong statistical dependencies: plausible combinations of features are not arbitrary.
2. Learning only individual marginals is generally insufficient:

$$p_{\text{data}}(\boldsymbol{x}) \neq \prod_{d=1}^{D} p_{\text{data}}(x_d).$$

3. A useful model captures joint structure, including correlations, constraints, and multiple modes.
4. Successful generation means learning the distribution, not memorizing a finite set of examples.

Maximum likelihood and KL divergence

1. One general principle is maximum likelihood estimation:

$$\theta^* = \arg\max_\theta \sum_{n=1}^{N} \log p_\theta(\mathrm{x}^{(n)}).$$

2. In the population limit, this is equivalent to minimizing cross-entropy:

$$\min_\theta \mathbb{E}_{\mathrm{x}\sim p_{\mathrm{data}}}[-\log p_\theta(\mathrm{x})].$$

3. Forward KL divergence can be written as the difference between cross-entropy and entropy:

$$D_{\mathrm{KL}}(p_{\mathrm{data}} \| p_\theta) = \underbrace{H(p_{\mathrm{data}}, p_\theta)}_{\text{Cross-entropy}} - \underbrace{H(p_{\mathrm{data}})}_{\text{Entropy (constant in } \theta)}.$$

   Thus, minimizing cross-entropy is equivalent to minimizing forward KL divergence.

4. Some generative methods instead optimize tractable surrogate objectives.

Three distinct computational questions

1. **Sampling:** Can we efficiently generate $\boldsymbol{x} \sim p_\theta$?
2. **Density evaluation:** Can we compute $p_\theta(\boldsymbol{x})$ or $\log p_\theta(\boldsymbol{x})$ for a given observation?
3. **Learning:** Can we optimize a useful objective from samples of p_{data}?

These properties need not come together. A model can sample efficiently even when its likelihood is difficult to evaluate, or evaluate likelihood efficiently while requiring sequential generation.

Two complementary modeling strategies

1. **Autoregressive factorization:** represent a joint distribution through a sequence of conditional distributions:

$$p_\theta(\boldsymbol{x}) = \prod_{d=1}^{D} p_\theta(x_d \mid x_1, \ldots, x_{d-1}).$$

2. **Latent-variable construction:** first sample a hidden variable $\boldsymbol{z}$, then generate an observation:

$$p_\theta(\boldsymbol{x}) = \int p_\theta(\boldsymbol{x} \mid \boldsymbol{z}) p_\theta(\boldsymbol{z})\, d\boldsymbol{z}.$$

3. These are not mutually exclusive: conditional distributions within a latent-variable model can themselves be autoregressive.

Autoregressive models: why the factorization works

1. The probability chain rule gives an exact identity for any joint distribution:
$$p(\boldsymbol{x}) = p(x_1)p(x_2 \mid x_1) \cdots p(x_D \mid x_1, \ldots, x_{D-1}).$$
2. A neural network parameterizes these conditionals; Transformer language models are a prominent example.
3. Maximum likelihood decomposes into a sum of conditional log-likelihoods:
$$\log p_\theta(\boldsymbol{x}) = \sum_{d=1}^{D} \log p_\theta(x_d \mid \boldsymbol{x}_{<d}).$$

Autoregressive models: training and sampling

1. During training, observed prefixes are available, so many conditional predictions can be evaluated in parallel.
2. During generation, each new coordinate or token depends on previously generated values:

$$x_1 \sim p_\theta(x_1), \quad x_d \sim p_\theta(x_d \mid \boldsymbol{x}_{<d}).$$

3. **Strength:** normalized likelihood and direct maximum likelihood training are often tractable.
4. **Tradeoff:** sampling is sequential in the chosen ordering, although global dependencies can be modeled through long-range context.

Latent-variable models: a generative mechanism

1. Introduce an unobserved variable $\boldsymbol{z}$ to represent underlying factors of variation.
2. Specify a prior and a conditional observation model:

$$\boldsymbol{z} \sim p_\theta(\boldsymbol{z}), \qquad \boldsymbol{x} \mid \boldsymbol{z} \sim p_\theta(\boldsymbol{x} \mid \boldsymbol{z}).$$

3. Together they define the joint distribution:

$$p_\theta(\boldsymbol{x}, \boldsymbol{z}) = p_\theta(\boldsymbol{x} \mid \boldsymbol{z}) p_\theta(\boldsymbol{z}).$$

4. The observable distribution is obtained by marginalizing the latent variable.

Why introduce latent variables?

1. Latent variables provide a mechanism for expressing complex dependence among observed dimensions.
2. For example, even if observed coordinates are conditionally independent given $\boldsymbol{z}$,

$$p_\theta(\boldsymbol{x} \mid \boldsymbol{z}) = \prod_{d=1}^{D} p_\theta(x_d \mid \boldsymbol{z}),$$

   they need not be independent after integrating out $\boldsymbol{z}$.
3. Shared latent factors can represent global structure, variability, and known physical or semantic constraints.
4. Interpretability is possible but not guaranteed by the presence of latent variables alone.

Why can latent-variable learning be difficult?

1. To evaluate the probability of an observation, we must integrate over all possible hidden explanations:

$$p_\theta(\boldsymbol{x}) = \int p_\theta(\boldsymbol{x}, \boldsymbol{z})\, d\boldsymbol{z}.$$

2. Maximum likelihood therefore involves a log-integral:

$$\log p_\theta(\boldsymbol{x}) = \log \int p_\theta(\boldsymbol{x}, \boldsymbol{z})\, d\boldsymbol{z}.$$

3. This integral can be computationally intractable.
4. Restricted invertible constructions, exact inference, or approximation methods address this difficulty in different ways.

A broader landscape of generative models

1. **Autoregressive models:** Factorize the joint distribution into sequential conditional probabilities.
2. **Variational autoencoders (VAEs):** Learn latent variable models using variational inference and the ELBO.
3. **Generative adversarial networks (GANs):** Learn to generate realistic samples through adversarial training.
4. **Normalizing flows:** Transform a simple distribution using invertible mappings with tractable likelihoods.
5. **Diffusion and score-based models:** Learn reverse denoising dynamics or score fields for generative sampling.
6. **Flow matching:** Learn velocity fields that transport a simple distribution toward the data distribution.

Focus of these materials: Flow matching, diffusion models, and score matching, including their mathematical foundations and connections.

Outline

1 Introduction

2 Flow Matching
- Mathematical Foundations
- Basic Generation Paradigm
- Training and Sampling
- Theoretical Analysis
- Rectified Flow and Optimal Transport
- Riemannian Flow Matching
- Discrete Flow Matching

3 Diffusion Models

4 Score Matching

5 Connections

6 Frontiers

7 Latent Variable Models

8 Variational Inference

Scalar fields and vector fields

1. Consider a point in d-dimensional space:

$$\mathrm{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d.$$

2. A scalar field assigns a scalar to every point:

$$f : \mathbb{R}^d \to \mathbb{R}.$$

   For example, in fluid mechanics, density is a scalar field: $\rho(\mathrm{x}, t)$.

3. In flow matching, the probability density is a scalar field: $p_t(\mathrm{x})$.

4. A vector field assigns a vector to every point:

$$\mathrm{v} : \mathbb{R}^d \to \mathbb{R}^d.$$

   For example, the velocity field is $\mathrm{v}(\mathrm{x}, t) = \begin{pmatrix} v_1(\mathrm{x}, t) \\ \vdots \\ v_d(\mathrm{x}, t) \end{pmatrix}$.

5. In fluid mechanics, velocity describes motion in physical 3D space.

6. In flow matching, velocity describes motion in the data space, which can have any dimension.

Gradient

1. The gradient of a scalar field is a vector:

$$\boxed{\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_d} \end{pmatrix}.}$$

2. The gradient points in the direction of the steepest increase of f.
3. For example, let

$$f(x, y) = x^2 + y^2.$$

4. Then

$$\nabla f = \begin{pmatrix} 2x \\ 2y \end{pmatrix}.$$

5. For example, in score matching, the score function is the gradient of the log-probability density:

$$\boxed{\mathrm{s}_t(\mathrm{x}) = \nabla \log p_t(\mathrm{x}).}$$

Divergence

1. The divergence of a vector field is a scalar:

$$\boxed{\nabla \cdot \mathbf{v} = \sum_{i=1}^{d} \frac{\partial v_i}{\partial x_i}.}$$

2. In 3D physical space,

$$\nabla \cdot \mathbf{v} = \frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z}.$$

3. Intuitively, divergence measures the local tendency of a vector field to spread outward or converge inward.
4. For incompressible fields (constant density), the divergence is zero.

Divergence

1. For example, consider
$$\mathbf{v}(x, y) = \begin{pmatrix} x \\ y \end{pmatrix}.$$
2. Then
$$\nabla \cdot \mathbf{v} = \frac{\partial x}{\partial x} + \frac{\partial y}{\partial y} = 2.$$
The field expands outward.
3. In contrast, if
$$\mathbf{v}(x, y) = \begin{pmatrix} -x \\ -y \end{pmatrix},$$
then
$$\nabla \cdot \mathbf{v} = -2.$$
The field contracts toward the origin.
4. Note that zero divergence does not necessarily mean zero velocity.
5. For example, uniform translation has zero divergence.

Flux: how much flows through a surface?

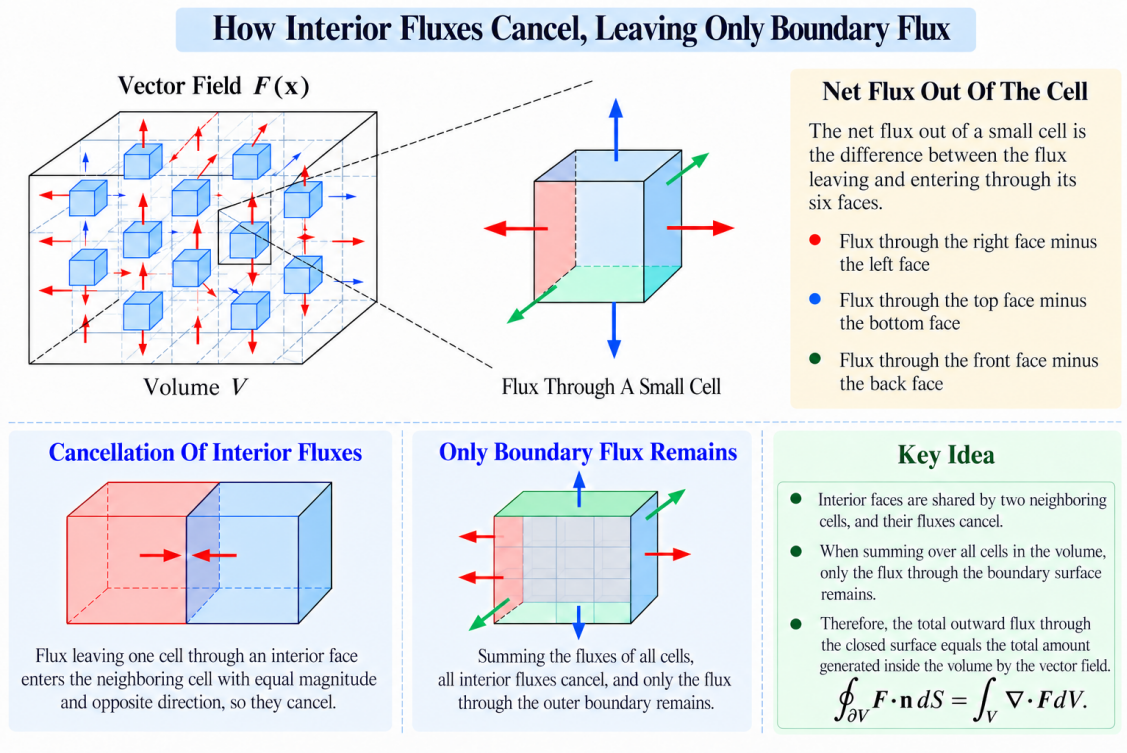
1. Suppose a fluid has density $\rho(\mathrm{x}, t)$ and velocity $\mathrm{v}(\mathrm{x}, t)$.
2. Define the mass flux
$$\boxed{\mathrm{J} = \rho \mathrm{v}.}$$
3. The mass flowing outward through a surface S, per unit time, is
$$\boxed{\int_S \rho \mathrm{v} \cdot \mathrm{n}\, dS,}$$
where n is the outward unit normal.
4. The dot product
$$\mathrm{v} \cdot \mathrm{n}$$
extracts the component of velocity perpendicular to the surface.
5. In flow matching, we replace mass density by probability density:
$$\boxed{\mathrm{J}_t(\mathrm{x}) = p_t(\mathrm{x})\mathrm{v}_t(\mathrm{x}).}$$
This is the probability flux.
6. It describes how probability mass moves through space.
7. **Divergence of flux measures the net outflow.**

The divergence theorem

1. The divergence theorem connects the total outward flux through the boundary of a region to the divergence inside the region:

$$\int_{\partial\Omega} \mathsf{J} \cdot \mathsf{n}\, dS = \int_{\Omega} \nabla \cdot \mathsf{J}\, d\mathsf{x}.$$

   Here Ω is a region and $\partial\Omega$ is its boundary.
2. Outward flux through boundary = total divergence inside region.



Deriving the continuity equation from conservation of mass

1. Consider a fixed region Ω containing fluid. The total mass inside is

$$M(t) = \int_\Omega \rho(\mathsf{x}, t)\, d\mathsf{x}.$$

2. Conservation of mass says

$$\underbrace{\frac{dM(t)}{dt}}_{\text{Rate of mass change}} = -\underbrace{\int_{\partial\Omega} \rho \mathsf{v} \cdot \mathsf{n}\, dS}_{\text{Net outward mass flux}}.$$

3. Substituting the definition of $M(t)$,

$$\frac{d}{dt}\int_\Omega \rho\, d\mathsf{x} = -\int_{\partial\Omega} \rho \mathsf{v} \cdot \mathsf{n}\, dS.$$

4. For a fixed region, move the time derivative inside the integral:

$$\int_\Omega \frac{\partial \rho}{\partial t}\, d\mathsf{x} = -\int_{\partial\Omega} \rho \mathsf{v} \cdot \mathsf{n}\, dS.$$

Deriving the continuity equation from conservation of mass

1. Apply the divergence theorem:

$$\int_{\Omega} \left[\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathsf{v}) \right] d\mathsf{x} = 0.$$

2. Since this holds for every arbitrary region Ω, assuming sufficient regularity,

$$\boxed{\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathsf{v}) = 0.}$$

Laplacian as the divergence of a gradient

1. For a scalar field $\phi(\mathrm{x})$, the Laplacian is the divergence of its gradient:

$$\boxed{\Delta\phi = \nabla^2\phi = \nabla\cdot(\nabla\phi) = \frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2} + \frac{\partial^2\phi}{\partial z^2}.}$$

2. The gradient $\nabla\phi$ points toward the steepest increase of ϕ and can be viewed as a vector flux field.
3. The divergence $\nabla\cdot(\nabla\phi)$ measures the local net outward flux of this gradient field:
   - $\Delta\phi > 0$: net outward gradient flux.
   - $\Delta\phi < 0$: net inward gradient flux.
4. For diffusion, the physical flux is $\mathrm{J} = -D\nabla\phi$. Conservation gives

$$\partial_t\phi = -\nabla\cdot\mathrm{J} = D\Delta\phi.$$

From fluid mechanics to flow matching

Fluid Mechanics	**Flow Matching**
Mass density $\rho(\mathrm{x}, t)$	Probability density $p_t(\mathrm{x})$
Fluid velocity v	Generative velocity v_t
Mass flux $\rho\mathrm{v}$	Probability flux $p_t\mathrm{v}_t$
Conservation of mass	Conservation of probability

1. Replace ρ with p_t:

$$\boxed{\frac{\partial p_t}{\partial t} + \nabla \cdot (p_t \mathrm{v}_t) = 0.}$$

2. The interpretation is:
   - $\partial p_t / \partial t$: How rapidly probability density changes at a fixed location.
   - $\nabla \cdot (p_t \mathrm{v}_t)$: Net outward probability flux per unit volume.
   - Their sum is zero because probability is conserved.

Summary

Gradient:	∇f
Divergence:	$\nabla \cdot \mathsf{v}$
Laplacian:	$\Delta f = \nabla \cdot (\nabla f)$
Probability flux:	$\mathsf{J}_t = p_t \mathsf{v}_t$
Continuity equation:	$\dfrac{\partial p_t}{\partial t} + \nabla \cdot (p_t \mathsf{v}_t) = 0$

1. The essential physical intuition is that probability behaves like a conserved fluid.
2. A velocity field moves probability mass around, changing its local density without creating or destroying total probability.
3. The Laplacian is the divergence of the gradient: it measures the local net outward flux of the gradient field and appears in the diffusion term of the Fokker–Planck equation.

Terminology

1. **Objects to Generate:** We focus on vectors z representing data objects (e.g., images, videos).
2. **Data Distribution:** Distribution that places higher probability on objects that we consider "good".
3. **Generation as Sampling:** Generating an object is equivalent to sampling from the data distribution.
4. **Dataset:** Finite number of samples from the data distribution used for training.
5. **Conditional Generation:** Condition on label y and sample from the conditional data distribution.
6. **Generative Model:** Train a model to transform samples from a simple (e.g., Gaussian) distribution into the data distribution.

From discrete transformations to continuous flows

1. In a discrete generative model (e.g., diffusion model), we might use a sequence of transformations:

$$x_0 \to x_1 \to \cdots \to x_T.$$

2. In flow matching, we instead introduce a continuous trajectory

$$x_t, \qquad t \in [0, 1].$$

3. The trajectory satisfies an ordinary differential equation:

$$\boxed{\frac{dx_t}{dt} = v_t(x_t).}$$

   Here

$$v_t : \mathbb{R}^d \to \mathbb{R}^d$$

   is a time-dependent vector field.

4. The vector field specifies the instantaneous velocity of a point at position x and time t.
5. A solution to an ODE is defined by a trajectory.

The basic generative modeling problem

1. Suppose we have samples from an unknown data distribution

$$x_1 \sim p_{\text{data}}(x).$$

2. Our goal is to construct a generative model capable of producing new samples from this distribution.
3. For example, p_{data} could represent the distribution of natural images, molecular configurations, or high-dimensional physical states.
4. Rather than learning p_{data} explicitly, flow matching learns how to transform samples from a simple distribution into samples from the data distribution.
5. We start with

$$\boxed{x_0 \sim p_0(x) = \mathcal{N}(0, I).}$$

6. We want to transform x_0 continuously into a sample

$$x_1 \sim p_1(x) = p_{\text{data}}(x).$$

Here $t \in [0, 1]$ denotes continuous time.

Examples

1. For example, in one dimension, suppose

$$v_t(x) = 2.$$

Then

$$\frac{dx_t}{dt} = 2,$$

so

$$x_t = x_0 + 2t.$$

2. Every point moves to the right with constant velocity.

Examples

1. A more interesting example is

$$v_t(x) = -x.$$

   Then

$$\frac{dx_t}{dt} = -x_t,$$

   whose solution is

$$x_t = e^{-t} x_0.$$

2. Points move toward the origin, and the distribution contracts.
3. Thus, a vector field determines how individual samples move and, consequently, how their probability distribution evolves.

The flow map

1. Given the ODE
$$\frac{dx_t}{dt} = v_t(x_t),$$
define the flow map
$$\phi_t(x_0) = x_t,$$
where x_t is the solution starting from x_0.
2. More explicitly,
$$\boxed{\frac{d}{dt}\phi_t(x) = v_t(\phi_t(x)), \qquad \phi_0(x) = x.}$$
3. If the initial random variable is
$$X_0 \sim p_0,$$
then
$$X_t = \phi_t(X_0)$$
has some probability distribution p_t.

The flow map

1. We write
$$\boxed{p_t = (\phi_t)_{\#} p_0,}$$
where $\#$ denotes the pushforward of a probability distribution.
2. This notation simply means that the distribution p_0 is transformed by the flow map ϕ_t.
3. For example, if
$$X_0 \sim \mathcal{N}(0, 1)$$
and
$$X_t = 2X_0 + t,$$
then
$$X_t \sim \mathcal{N}(t, 4).$$
4. The distribution changes because the samples have been transformed.
5. The goal of generative modeling is to learn a flow satisfying
$$\boxed{(\phi_1)_{\#} p_0 = p_{\text{data}}.}$$
6. Takeaway: Vector fields define ODEs whose solutions are flow maps.

Euler method for Simulating an ODE

1. Recall the flow ODE and its initial condition:

$$\frac{d}{dt}X_t = u_t(X_t), \qquad X_0 = x_0, \qquad X_t = \psi_t(x_0).$$

2. Discretize time into N steps:

$$t_k = kh, \qquad h = \frac{1}{N}, \qquad k = 0, \ldots, N.$$

3. Using a first-order Taylor approximation:

$$X_{t_{k+1}} \approx X_{t_k} + h\, u_{t_k}(X_{t_k}).$$

4. The Euler method approximates the flow iteratively:

$$\boxed{x_{k+1} = x_k + h\, u_{t_k}(x_k), \qquad x_0 = X_0.}$$

5. For flow matching, replace u_t with the learned velocity field u_t^θ and sample $x_0 \sim p_0$. The final state x_N approximates a sample from p_1.

The continuity equation: how distributions evolve

1. An ODE describes the motion of individual samples.
2. We also need an equation describing the evolution of their probability distribution.
3. This is given by the continuity equation:

$$\boxed{\frac{\partial p_t(\mathrm{x})}{\partial t} + \nabla \cdot (p_t(\mathrm{x})\mathrm{v}_t(\mathrm{x})) = 0.}$$

4. The same structure as the conservation of mass in fluid mechanics.

The continuity equation: how distributions evolve

1. Define the probability flux

$$\mathrm{J}_t(\mathrm{x}) = p_t(\mathrm{x})\mathrm{v}_t(\mathrm{x}).$$

   Then

$$\frac{\partial p_t}{\partial t} + \nabla \cdot \mathrm{J}_t = 0.$$

2. - $p_t(\mathrm{x})$ is probability density.
   - $\mathrm{v}_t(\mathrm{x})$ is velocity.
   - $\mathrm{J}_t(\mathrm{x})$ is probability flux.
3. The continuity equation expresses conservation of probability.
4. The continuity equation describes how a probability density evolves under a velocity field.
5. It does not by itself specify which velocity field should be used.

What is a probability path?

1. Suppose we have two distributions:

$$p_0(x) = \mathcal{N}(0, I)$$

$$p_1(x) = p_{\text{data}}(x).$$

2. A probability path is a family of distributions that connects p_0 and p_1.

$$\boxed{\{p_t(x)\}_{t\in[0,1]}}$$

3. For example, consider a simple one-dimensional probability path:

$$\boxed{p_t(x) = \mathcal{N}(2t, 1).}$$

4. At different times,

$$p_0(x) = \mathcal{N}(0, 1),$$

$$p_{0.5}(x) = \mathcal{N}(1, 1),$$

$$p_1(x) = \mathcal{N}(2, 1).$$

5. The probability path tells us what distribution we want at each time, but it does not directly tell us how individual points should move.
6. That is the role of the vector field.

A vector field to generate a probability path

1. Suppose we define a time-dependent velocity field
$$v_t(x).$$
2. Each sample moves according to the ODE
$$\boxed{\frac{dx_t}{dt} = v_t(x_t).}$$
3. If we initially sample
$$X_0 \sim p_0,$$
and evolve every sample using this ODE, we obtain random variables X_t.
4. We say that the vector field generates the probability path p_t if
$$\boxed{X_t \sim p_t \qquad \text{for every } t \in [0,1].}$$
5. Equivalently, the velocity field must satisfy the continuity equation:
$$\boxed{\frac{\partial p_t}{\partial t} + \nabla \cdot (p_t v_t) = 0.}$$

Example

1. Consider
$$p_t(x) = \mathcal{N}(2t, 1).$$
2. We want to find a velocity field $v_t(x)$ that generates this path.
3. A natural choice is
$$\boxed{v_t(x) = 2.}$$
4. This means every point moves to the right at speed 2.
5. The ODE becomes
$$\frac{dx_t}{dt} = 2.$$
6. Integrating,
$$\boxed{x_t = x_0 + 2t.}$$
7. Since
$$X_0 \sim \mathcal{N}(0, 1),$$
we obtain
$$X_t = X_0 + 2t \sim \mathcal{N}(2t, 1).$$
8. The vector field $v_t(x) = 2$ generates the probability path $p_t(x) = \mathcal{N}(2t, 1)$.

One-sentence summary

Learning a vector field that generates a probability path means learning a velocity function such that, when samples from the initial distribution move according to its ODE, their distribution at every time t matches the prescribed p_t.

What exactly does flow matching learn?

1. Suppose we choose a family of probability distributions

$$p_t(x), \qquad t \in [0,1],$$

such that

$$p_0 = \mathcal{N}(0, I), \qquad p_1 = p_{\text{data}}.$$

2. Learn a vector field v_t that generates this probability path.
3. Let the neural network be

$$v_\theta(x, t),$$

where θ denotes its trainable parameters.
4. If the desired velocity field $u_t(x)$ were known, we could minimize

$$\boxed{\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{t\sim\mathcal{U}[0,1],\, x\sim p_t}\left[\|v_\theta(x,t) - u_t(x)\|^2\right].}$$

5. It is simply a regression problem:
   - Input: position x and time t.
   - Target: desired velocity $u_t(x)$.

Marginal and conditional probability path

1. For a complicated data distribution, the marginal probability path p_t and its corresponding velocity field u_t may be difficult to compute.
2. The central innovation of flow matching is to avoid having to compute this marginal velocity field explicitly.
3. The key idea is to construct simple probability paths conditioned on individual data samples, rather than directly constructing the complicated marginal probability path.
4. A given probability path does not necessarily determine a unique velocity field.
5. The continuity equation constrains

$$\nabla \cdot (p_t v_t),$$

   rather than uniquely specifying v_t.
6. Flow matching chooses a particular field through its construction of conditional paths and its regression objective.

Conditional flow matching

1. Let
$$x_1 \sim p_{\text{data}}$$
be a random data sample.
2. For each fixed x_1, construct a conditional distribution
$$p_t(x \mid x_1).$$
3. The marginal distribution at time t is
$$\boxed{p_t(x) = \int p_t(x \mid x_1) p_{\text{data}}(x_1)\, dx_1.}$$
4. This is exactly the marginalization principle used in latent variable models.
5. Here, x_1 acts as the conditioning variable, and we integrate it out to obtain the marginal distribution of x_t.

Example

1. The important observation is that even if the marginal distribution p_t is complicated, each conditional distribution $p_t(\mathrm{x} \mid \mathrm{x}_1)$ can be simple.
2. For example, we can choose

$$\boxed{p_t(\mathrm{x} \mid \mathrm{x}_1) = \mathcal{N}\left(\mathrm{x}; t\mathrm{x}_1, (1-t)^2 \mathrm{I}\right).}$$

3. This conditional distribution has the desired endpoint behavior:
   At $t = 0$,
   $$p_0(\mathrm{x} \mid \mathrm{x}_1) = \mathcal{N}(0, \mathrm{I}).$$
   At $t = 1$, it becomes a point mass at x_1:
   $$p_1(\mathrm{x} \mid \mathrm{x}_1) = \delta(\mathrm{x} - \mathrm{x}_1).$$
4. Therefore, after marginalization,
   $$p_0(\mathrm{x}) = \mathcal{N}(0, \mathrm{I}), \qquad p_1(\mathrm{x}) = p_{\mathrm{data}}(\mathrm{x}).$$
5. We have constructed a probability path connecting noise and data without needing an explicit formula for the data density.

Linear interpolation probability path

1. In the example, the probability path was simple enough that we could find the velocity field analytically.
2. But in generative modeling, p_{data} may be extremely complicated. Thus, learning is required.
3. We can prescribe a probability path using

$$\boxed{x_t = (1-t)x_0 + tx_1,}$$

where

$$x_0 \sim \mathcal{N}(0, I), \qquad x_1 \sim p_{\text{data}},$$

independently.

4. This is linear interpolation between a noise sample and a data sample.
5. This defines the distribution p_t of the random variable x_t, even though we generally cannot write down its density explicitly.

The simplest flow matching construction

1. Now differentiate with respect to t:

$$\begin{aligned}\frac{dx_t}{dt} &= \frac{d}{dt}\left[(1-t)x_0 + tx_1\right] \\ &= -x_0 + x_1.\end{aligned}$$

2. Therefore,

$$\boxed{\frac{dx_t}{dt} = x_1 - x_0.}$$

3. This gives an extremely simple training target.
4. The velocity along this interpolated path is constant:

$$\boxed{u_t = x_1 - x_0.}$$

5. This is the foundation of linear-interpolation conditional flow matching.
6. Important subtlety: this velocity is associated with a particular sampled pair (x_0, x_1).
7. It is not generally the same as the marginal velocity field learned by the neural network.

The training objective

1. Using linear interpolation, the training objective becomes

$$\mathcal{L}_{\mathrm{CFM}}(\theta) = \mathbb{E}_{\substack{t\sim\mathcal{U}[0,1] \\ x_0\sim p_0 \\ x_1\sim p_{\mathrm{data}}}} \left[\left\| v_\theta((1-t)x_0 + t x_1, t) - (x_1 - x_0) \right\|^2 \right].$$

2. To train the model, we only need:
   1. A sample x_1 from the dataset.
   2. A Gaussian noise sample x_0.
   3. A random time t.
   4. A linear interpolation x_t.
   5. A neural network predicting the velocity.
3. The target velocity is simply

$$x_1 - x_0.$$

4. No ODE needs to be solved during training.
5. No score function needs to be evaluated.
6. No explicit probability density needs to be computed.
7. This is why flow matching is often described as a simulation-free training method for continuous normalizing flows.

Flow Matching Training Procedure (General)

Algorithm Flow Matching Training Procedure (General)

Require: A dataset of samples $z \sim p_{\text{data}}$, neural network u^θ.

1: **for** each mini-batch of data **do**

2: Sample a data example z from the dataset.

3: Sample a random time $t \sim \text{Unif}[0, 1]$.

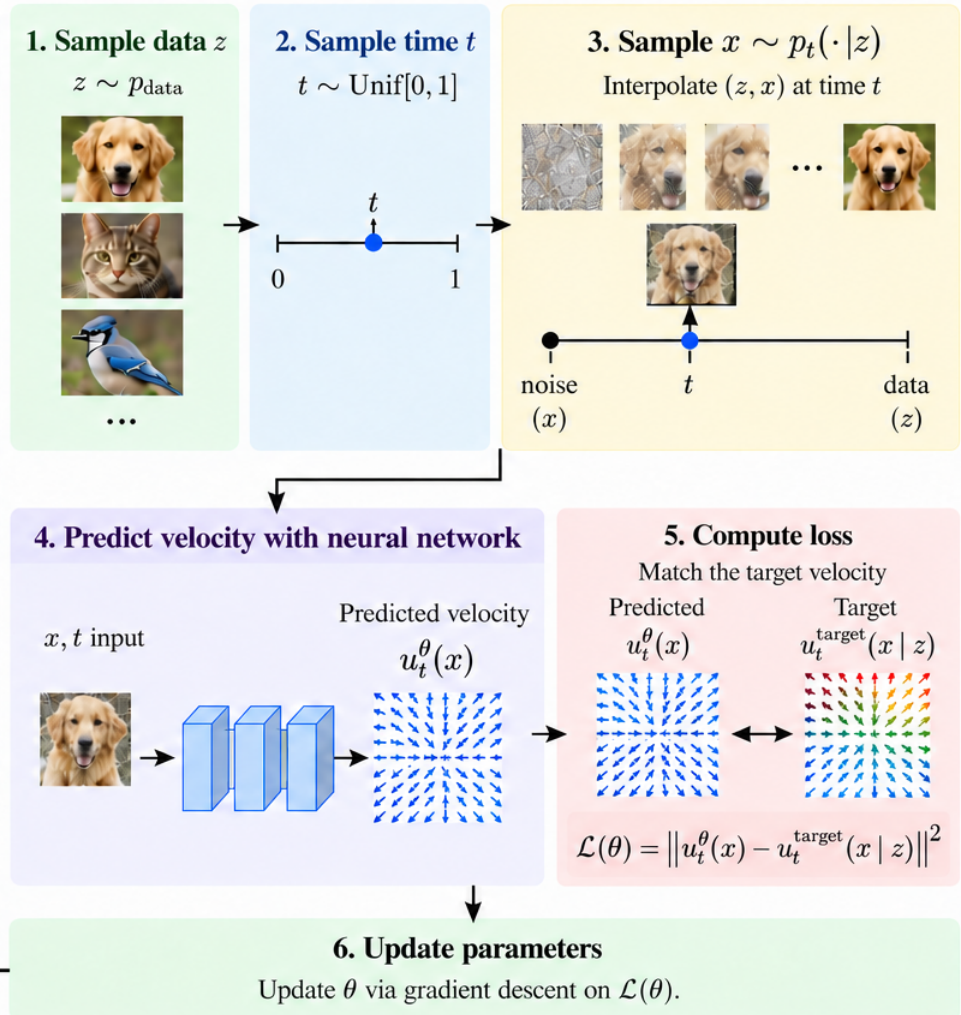
4: Sample $x \sim p_t(\cdot|z)$.

5: Compute loss

$$\mathcal{L}(\theta) = \left\| u_t^\theta(x) - u_t^{\text{target}}(x \mid z) \right\|^2$$

6: Update the model parameters θ via gradient descent on $\mathcal{L}(\theta)$.

7: **end for**



A concrete numerical example

1. Suppose
$$x_0 = \begin{pmatrix} 0 \\ 2 \end{pmatrix}, \qquad x_1 = \begin{pmatrix} 4 \\ 6 \end{pmatrix}.$$
Choose $t = 0.25$.
2. The interpolated point is
$$x_t = 0.75 \begin{pmatrix} 0 \\ 2 \end{pmatrix} + 0.25 \begin{pmatrix} 4 \\ 6 \end{pmatrix} = \begin{pmatrix} 1 \\ 3 \end{pmatrix}.$$
3. The target velocity is $u_t = x_1 - x_0 = \begin{pmatrix} 4 \\ 4 \end{pmatrix}$.
4. Suppose the neural network predicts
$$v_\theta(x_t, t) = \begin{pmatrix} 3.5 \\ 4.2 \end{pmatrix}.$$
5. Then the training loss for this sample is
$$\mathcal{L} = \left\| \begin{pmatrix} 3.5 \\ 4.2 \end{pmatrix} - \begin{pmatrix} 4 \\ 4 \end{pmatrix} \right\|^2 = (-0.5)^2 + (0.2)^2 = \boxed{0.29}.$$
6. The neural network parameters are updated to reduce this loss.

Generating new samples

1. Once the neural network has learned, we have
$$v_\theta(x, t),$$
2. First sample
$$x_0 \sim \mathcal{N}(0, I).$$
3. Then solve
$$\boxed{\frac{dx_t}{dt} = v_\theta(x_t, t), \qquad t \in [0, 1].}$$
4. The final state x_1 is the generated sample.
5. A simple numerical method is forward Euler:
$$\boxed{x_{t+\Delta t} = x_t + \Delta t\, v_\theta(x_t, t).}$$
6. For example, using ten steps,
$$\Delta t = 0.1.$$
7. The model repeatedly predicts a velocity and updates the sample.
8. Unlike the training process, sampling requires numerical integration of the learned dynamics.

Sampling procedure

Sampling from a Flow Model with Euler Method

Algorithm Sampling from a Flow Model with Euler Method

Require: Trained velocity field $v_\theta(x, t)$, number of steps n.

1: Set $t = 0$.

2: Set step size $\Delta t = \frac{1}{n}$.

3: Sample initial point $x_0 \sim p_{\text{init}}$. *(Random initialization)*

4: **for** $i = 0, \ldots, n-1$ **do**

5: $\quad x_{t+\Delta t} = x_t + \Delta t\, v_\theta(x_t, t)$

6: $\quad t \leftarrow t + \Delta t$

7: **end for**

8: **return** x_1 *(Generated sample)*

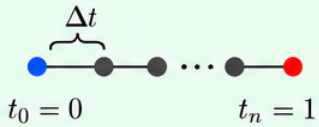
1. Time interval

$t \in [0, 1]$



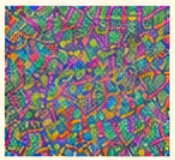
2. Step size

$\Delta t = \frac{1}{n}$



3. Initialize from noise

$x_0 \sim p_{\text{init}}$

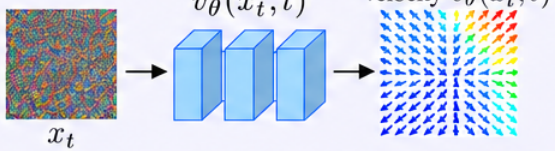


Initial sample x_0
(e.g., Gaussian noise)

Repeat n steps

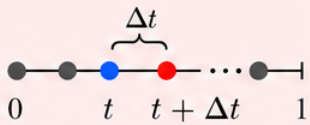
4. Euler update

$x_{t+\Delta t} = x_t + \Delta t\, v_\theta(x_t, t)$

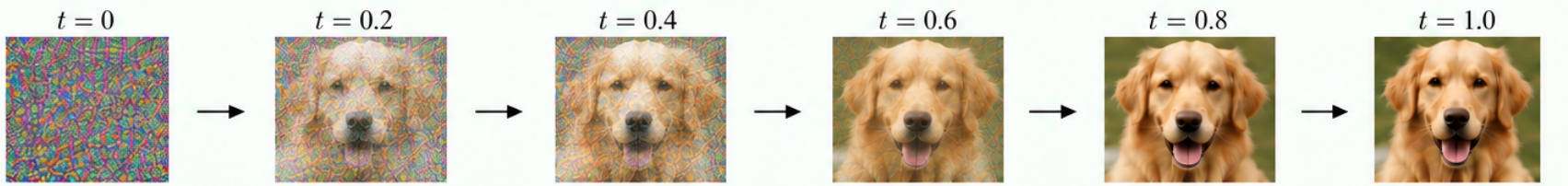


5. Update time

$t \leftarrow t + \Delta t$



6. Generated sample after n steps (e.g., $n = 5$, $\Delta t = 0.2$)



Generated sample x_1

A complete one-dimensional example

1. Suppose
$$X_0 \sim \mathcal{N}(0,1), \qquad X_1 \sim \mathcal{N}(2,1),$$
2. Define
$$X_t = (1-t)X_0 + tX_1.$$
3. Because linear combinations of independent Gaussian random variables are Gaussian,
$$\boxed{X_t \sim \mathcal{N}\left(2t, (1-t)^2 + t^2\right).}$$
4. Let
$$s_t^2 = (1-t)^2 + t^2.$$
5. The target velocity for each pair is
$$U = X_1 - X_0.$$

A complete one-dimensional example

1. We want
$$v^*(x,t) = \mathbb{E}[U \mid X_t = x].$$
2. Since U and X_t are jointly Gaussian, the conditional expectation is
$$\mathbb{E}[U \mid X_t = x] = \mathbb{E}[U] + \frac{\mathrm{Cov}(U, X_t)}{\mathrm{Var}(X_t)} \left(x - \mathbb{E}[X_t]\right).$$
3. Now
$$\mathbb{E}[U] = 2, \qquad \mathbb{E}[X_t] = 2t,$$
and
$$\begin{aligned}\mathrm{Cov}(U, X_t) &= \mathrm{Cov}\left(X_1 - X_0, (1-t)X_0 + tX_1\right) \\ &= t - (1-t) \\ &= 2t - 1.\end{aligned}$$
4. Therefore,
$$\boxed{v^*(x,t) = 2 + \frac{2t-1}{(1-t)^2 + t^2}(x - 2t).}$$
5. This is the exact marginal velocity field.

A complete one-dimensional example

1. Notice that it depends on both x and t, even though the velocity of every individual interpolated pair is constant.
2. The associated ODE has solution

$$\boxed{X_t = 2t + s_t X_0,}$$

   where $X_0 \sim \mathcal{N}(0, 1)$.
3. Indeed,

$$X_t \sim \mathcal{N}(2t, s_t^2).$$

   At $t = 1$,

$$X_1 = 2 + X_0 \sim \mathcal{N}(2, 1).$$

4. Thus, integrating the learned marginal velocity transports the source Gaussian into the desired target Gaussian.
5. This example also demonstrates that the ODE coupling between initial and final samples need not be the same as the independent coupling used to construct the training interpolations.

Why conditional flow matching works

1. We have two different objectives.
2. The ideal marginal flow matching objective is

$$\mathcal{L}_{\mathrm{FM}} = \mathbb{E}_{t,\mathsf{x}_t}\left[\|\mathsf{v}_\theta(\mathsf{x}_t, t) - \mathsf{u}_t(\mathsf{x}_t)\|^2\right].$$

3. The conditional objective is

$$\mathcal{L}_{\mathrm{CFM}} = \mathbb{E}_{t,\mathsf{x}_0,\mathsf{x}_1}\left[\|\mathsf{v}_\theta(\mathsf{x}_t, t) - (\mathsf{x}_1 - \mathsf{x}_0)\|^2\right].$$

4. Why should optimizing the second objective recover the velocity field needed for the first?
5. The answer comes from conditional expectation.

Review: theory of least squares regression

1. Given a joint distribution $p(\mathrm{x}, y)$, we seek a function $f : \mathbb{R}^d \to \mathbb{R}$ that minimizes the expected squared error:

$$\boxed{f^* = \arg\min_f \mathcal{L}(f), \qquad \mathcal{L}(f) = \mathbb{E}_{X,Y}[(Y - f(X))^2].}$$

2. By the law of total expectation (iterated expectation),

$$\begin{aligned}
\mathcal{L}(f) &= \int \int (y - f(\mathrm{x}))^2 p(\mathrm{x}, y)\, dy\, d\mathrm{x} \\
&= \int p(\mathrm{x}) \underbrace{\left[\int (y - f(\mathrm{x}))^2 p(y \mid \mathrm{x})\, dy \right]}_{\mathbb{E}_{Y|X=\mathrm{x}}[(Y - f(\mathrm{x}))^2]} d\mathrm{x} \\
&= \mathbb{E}_X \left[\mathbb{E}_{Y|X}[(Y - f(X))^2 \mid X] \right],
\end{aligned}$$

   where we used $p(\mathrm{x}, y) = p(y \mid \mathrm{x}) p(\mathrm{x})$.

3. The overall expected loss is the weighted average of the conditional losses at individual inputs x.

Review: theory of least squares regression

1. Define the conditional loss

$$\ell_{\mathrm{x}}(a) = \mathbb{E}[(Y - a)^2 \mid X = \mathrm{x}].$$

2. Then

$$\mathcal{L}(f) = \int p(\mathrm{x})\ell_{\mathrm{x}}(f(\mathrm{x}))\, d\mathrm{x}.$$

3. Because we optimize over all measurable functions f, the prediction $f(\mathrm{x})$ at one input does not constrain the prediction at another input. Moreover, the loss contains no terms coupling $f(\mathrm{x})$ and $f(\mathrm{x}')$.
4. Therefore, minimizing each conditional loss independently also minimizes their weighted average:

$$\boxed{f^*(\mathrm{x}) = \arg\min_a \mathbb{E}[(Y - a)^2 \mid X = \mathrm{x}].}$$

This holds almost everywhere with respect to $p(\mathrm{x})$.

Review: theory of least squares regression

1. Expanding the conditional loss,

$$\ell_x(a) = \mathbb{E}[Y^2 \mid X = x] - 2a\mathbb{E}[Y \mid X = x] + a^2,$$
$$\frac{d\ell_x(a)}{da} = -2\mathbb{E}[Y \mid X = x] + 2a = 0,$$
$$\frac{d^2\ell_x(a)}{da^2} = 2 > 0.$$

2. Hence,

$$\boxed{f^*(x) = \mathbb{E}[Y \mid X = x].}$$

3. Under squared error loss, the optimal unrestricted regression function is the conditional expectation of Y given $X = x$.
4. The independent minimization argument generally does not apply to a restricted function class, such as linear functions or a fixed neural network architecture, because the shared parameters couple predictions at different inputs.

The optimal predictor under squared error

1. In conditional flow matching, define

$$Y = X_1 - X_0$$

and

$$Z = (X_t, t).$$

2. The conditional flow matching objective is

$$\boxed{\mathcal{L}_{\mathrm{CFM}}(\theta) = \mathbb{E}\left[\|v_\theta(X_t, t) - (X_1 - X_0)\|^2\right].}$$

3. By the regression theorem, the optimal predictor is

$$\boxed{v^*(x, t) = \mathbb{E}[X_1 - X_0 \mid X_t = x].}$$

4. This is precisely the marginal velocity field associated with the chosen interpolation.

Conditional and marginal velocity fields

1. Conditional = Per single data point
2. Marginal = Across distribution of data points

Conditional Velocity	**Marginal Velocity**
$X_1 - X_0$	$\mathbb{E}[X_1 - X_0 \mid X_t = \mathrm{x}]$
Velocity of one sampled endpoint pair.	Average velocity of pairs passing through a given location at time t.

3. Different noise-data pairs can pass through the same location at the same time, but with different velocities.
4. The neural network is trained using conditional velocities, but its optimal prediction is the marginal velocity.
5. This is what makes conditional flow matching possible.

Conditional and marginal velocity fields

1. Different noise-data pairs can pass through the same location at the same time, but with different velocities.
2. For example, in one dimension, consider $t = 0.5$.

Pair	X_0	X_1	$X_{0.5}$	Velocity $X_1 - X_0$
A	0	2	1	$+2$
B	2	0	1	-2
C	-1	3	1	$+4$

3. All three paths pass through

$$X_{0.5} = 1,$$

   but their velocities differ.
4. A neural network

$$v_\theta(x, t)$$

   receives only the current position x and time t.
5. At the input $(1, 0.5)$, it cannot output three different velocities simultaneously.
6. What should it predict?

Conditional and marginal velocity fields

1. The answer is the conditional average velocity.
2. For these three equally weighted paths, that average would be

$$\frac{2 + (-2) + 4}{3} = \frac{4}{3}.$$

3. In the actual model, the average is taken over the full conditional distribution of possible endpoint pairs.

Why the same minimizer?

1. The ideal flow matching objective is

$$\boxed{\mathcal{L}_{\mathrm{FM}} = \mathbb{E}\left[\|\mathsf{v}_\theta(\mathsf{X}_t, t) - \mathsf{u}_t(\mathsf{X}_t)\|^2\right].}$$

2. The conditional objective is

$$\boxed{\mathcal{L}_{\mathrm{CFM}} = \mathbb{E}\left[\|\mathsf{v}_\theta(\mathsf{X}_t, t) - (\mathsf{X}_1 - \mathsf{X}_0)\|^2\right].}$$

   Here,

$$\mathsf{u}_t(\mathsf{x}) = \mathbb{E}[\mathsf{X}_1 - \mathsf{X}_0 \mid \mathsf{X}_t = \mathsf{x}].$$

3. We can show that the objectives differ only by a constant independent of θ.

Why the same minimizer?

1. Define
$$Y = X_1 - X_0, \qquad U = \mathbb{E}[Y \mid X_t, t].$$
2. Now write
$$v_\theta - Y = (v_\theta - U) + (U - Y).$$
3. Expanding the squared norm,
$$\begin{aligned} \|v_\theta - Y\|^2 = \|v_\theta - U\|^2 \\ + \|U - Y\|^2 \\ + 2(v_\theta - U)^\top (U - Y). \end{aligned}$$
4. Take expectations. The cross term vanishes because
$$\mathbb{E}[U - Y \mid X_t, t] = 0.$$

Why the same minimizer?

1. Therefore,

$$\boxed{\mathcal{L}_{\mathrm{CFM}}(\theta) = \mathcal{L}_{\mathrm{FM}}(\theta) + \underbrace{\mathbb{E}[\|\mathsf{Y} - \mathsf{U}\|^2]}_{\text{Independent of } \theta}.}$$

2. Consequently,

$$\boxed{\nabla_\theta \mathcal{L}_{\mathrm{CFM}} = \nabla_\theta \mathcal{L}_{\mathrm{FM}}.}$$

   And hence,

$$\boxed{\arg\min_\theta \mathcal{L}_{\mathrm{CFM}} = \arg\min_\theta \mathcal{L}_{\mathrm{FM}}.}$$

3. The two objectives generally have different numerical loss values, but their gradients and minimizers are the same.
4. This equality assumes the same sampling distribution and a fixed interpolation construction.

Why the learned velocity field works

1. Consider the linear interpolation

$$X_t = (1-t)X_0 + tX_1, \qquad t \in [0,1],$$

$X_0 \sim p_0$ and $X_1 \sim p_1$. Let p_t denote the probability distribution of X_t.

2. Define the marginal velocity field

$$\boxed{v^*(x,t) = \mathbb{E}[X_1 - X_0 \mid X_t = x].}$$

3. **Result 1: Continuity Equation.**
The marginal velocity field satisfies

$$\boxed{\frac{\partial p_t}{\partial t} + \nabla \cdot (p_t v^*) = 0.}$$

4. **Result 2: Probability Transport.**
Under suitable regularity and uniqueness conditions, the ODE

$$\boxed{\frac{dx_t}{dt} = v^*(x_t,t), \qquad x_0 \sim p_0}$$

generates the prescribed probability path p_t.

Proof: Continuity equation

1. For any smooth, compactly supported test function φ, using the chain rule and conditional expectation,

$$\begin{aligned}\frac{d}{dt}\mathbb{E}[\varphi(X_t)] &= \mathbb{E}\left[\nabla\varphi(X_t)^\top \frac{dX_t}{dt}\right] \\ &= \mathbb{E}\left[\nabla\varphi(X_t)^\top (X_1 - X_0)\right] \\ &= \mathbb{E}\left[\nabla\varphi(X_t)^\top v^*(X_t, t)\right] \\ &= \int p_t(x)\nabla\varphi(x)^\top v^*(x,t)\,dx \\ &= -\int \varphi(x)\nabla\cdot(p_t v^*)\,dx.\end{aligned}$$

2. On the other hand,

$$\frac{d}{dt}\mathbb{E}[\varphi(X_t)] = \int \varphi(x)\frac{\partial p_t}{\partial t}\,dx.$$

Proof: continuity equation

1. Equating the two expressions,

$$\int \varphi(\mathrm{x}) \left[\frac{\partial p_t}{\partial t} + \nabla \cdot (p_t \mathrm{v}^*) \right] d\mathrm{x} = 0.$$

2. Since φ is arbitrary,

$$\boxed{\frac{\partial p_t}{\partial t} + \nabla \cdot (p_t \mathrm{v}^*) = 0.}$$

Proof: probability transport

1. Consider the ODE
$$\frac{dx_t}{dt} = v^*(x_t, t), \qquad x_0 \sim p_0.$$
2. Let q_t denote the distribution induced by this ODE.
3. By the standard correspondence between ODE flows and the continuity equation,
$$\frac{\partial q_t}{\partial t} + \nabla \cdot (q_t v^*) = 0.$$
4. From Result 1, the prescribed path p_t satisfies
$$\frac{\partial p_t}{\partial t} + \nabla \cdot (p_t v^*) = 0.$$
5. Since p_t and q_t satisfy the same continuity equation with the same initial distribution, uniqueness of the solution implies
$$\boxed{q_t = p_t, \qquad \forall t \in [0, 1].}$$
6. Thus, the ODE generates the desired probability path.

An important subtlety

1. Straight conditional paths do not guarantee straight generated trajectories
2. For each sampled pair,

$$x_t = (1 - t)x_0 + tx_1$$

   is a straight line.
3. However, the learned velocity is

$$v^*(x, t) = \mathbb{E}[X_1 - X_0 \mid X_t = x].$$

4. When many interpolated paths overlap, their velocities are averaged.
5. Consequently, the ODE trajectories generated by the learned marginal vector field can be curved.

An important subtlety

1. Consider again the three velocities $+2, -2, +4$ passing through the same location.
2. The network predicts their conditional average, not one of the three velocities.
3. However, once the network is trained, a generated sample follows the deterministic ODE
$$\frac{dx_t}{dt} = v_\theta(x_t, t).$$
4. It does not randomly choose one of the original noise-data pairs at every time.
5. In other words:
   - Training uses randomly sampled straight-line paths.
   - The network learns their conditional mean velocity.
   - Generation follows trajectories of the learned marginal vector field.
   - Those generated trajectories need not be straight lines.

Key equations

1. Generative ODE

$$\boxed{\frac{dx_t}{dt} = v_\theta(x_t, t)}$$

2. Continuity equation

$$\boxed{\frac{\partial p_t}{\partial t} + \nabla \cdot (p_t v_t) = 0}$$

3. Linear interpolation

$$\boxed{x_t = (1-t)x_0 + t x_1}$$

4. Conditional flow matching objective

$$\boxed{\mathcal{L}_{\mathrm{CFM}} = \mathbb{E}\left[\|v_\theta(x_t, t) - (x_1 - x_0)\|^2\right]}$$

5. Optimal marginal velocity

$$\boxed{v_t^*(x) = \mathbb{E}[X_1 - X_0 \mid X_t = x]}$$

From flow matching to rectified flow

1. Recall the linear interpolation used in flow matching:

$$x_t = (1-t)x_0 + tx_1, \qquad t \in [0,1].$$

2. Each sampled pair follows a straight path with constant velocity:

$$\frac{dx_t}{dt} = x_1 - x_0.$$

3. The optimal marginal velocity field is

$$v^*(x,t) = \mathbb{E}\left[X_1 - X_0 \mid X_t = x\right].$$

4. The learned ODE follows this marginal field:

$$\frac{dx_t}{dt} = v_\theta(x_t, t).$$

5. Although the interpolated training paths are straight, the trajectories of the learned ODE need not be straight.

Couplings between probability distributions

1. A coupling specifies a joint distribution $\pi(x_0, x_1)$ with marginals p_0 and p_1:

$$\int \pi(x_0, x_1)\, dx_1 = p_0(x_0),$$

$$\int \pi(x_0, x_1)\, dx_0 = p_1(x_1).$$

2. Standard flow matching commonly uses independent sampling:

$$\pi(x_0, x_1) = p_0(x_0)p_1(x_1).$$

3. Other couplings can pair source and target samples differently while preserving both marginals.
4. The choice of coupling affects the interpolated probability path and the learned velocity field.
5. This raises a natural question: can we choose couplings that lead to more efficient transport?

Optimal transport and transport cost

1. Optimal transport seeks a coupling that minimizes the expected cost of moving probability mass.
2. For quadratic transport cost, the problem is

$$\min_{\pi \in \Pi(p_0, p_1)} \mathbb{E}_{(X_0, X_1) \sim \pi} \left[\|X_1 - X_0\|^2 \right],$$

   where $\Pi(p_0, p_1)$ denotes all couplings with marginals p_0 and p_1.
3. The minimum defines the squared 2-Wasserstein distance:

$$W_2^2(p_0, p_1) = \min_{\pi \in \Pi(p_0, p_1)} \mathbb{E}_{\pi} \left[\|X_1 - X_0\|^2 \right].$$

4. An optimal coupling minimizes the average squared displacement between endpoints.
5. Independent coupling is generally not optimal for this objective.

Rectified flow

1. Rectified flow learns a velocity field using the same linear interpolation:

$$x_t = (1 - t)x_0 + tx_1.$$

2. Its regression objective is

$$\mathcal{L}_{\mathrm{RF}}(\theta) = \mathbb{E}_{\substack{t\sim\mathcal{U}[0,1] \\ (x_0,x_1)\sim\pi}} \left[\|v_\theta(x_t, t) - (x_1 - x_0)\|^2 \right].$$

3. The learned ODE defines a new coupling between its initial and final states:

$$\frac{dx_t}{dt} = v_\theta(x_t, t), \qquad x_0 \sim p_0.$$

4. With an exact velocity field and suitable regularity conditions, the ODE preserves the prescribed marginal distributions.
5. The distinctive idea of rectified flow is to reuse the resulting coupling to improve the transport trajectories.

Reflow: straightening transport trajectories

1. Start with an initial coupling $(X_0, X_1) \sim \pi$ and train a rectified flow.
2. Generate new endpoint pairs by solving the learned ODE:

$$X'_0 \sim p_0, \qquad \frac{dX'_t}{dt} = v_\theta(X'_t, t), \qquad X'_1 = X'_{t=1}.$$

3. Use the new endpoint pairs to construct straight interpolation paths:

$$\widetilde{X}_t = (1-t)X'_0 + tX'_1.$$

4. Retrain the velocity field using $X'_1 - X'_0$ as the target.
5. This procedure is called reflow. It can reduce trajectory curvature and make numerical sampling more efficient.

Why rectification reduces transport cost

1. Consider the exact marginal velocity field

$$v^*(x, t) = \mathbb{E}[X_1 - X_0 \mid X_t = x].$$

2. By Jensen's inequality,

$$\begin{aligned}\mathbb{E}\left[\|v^*(X_t, t)\|^2\right] &= \mathbb{E}\left[\|\mathbb{E}[X_1 - X_0 \mid X_t]\|^2\right] \\ &\leq \mathbb{E}\left[\|X_1 - X_0\|^2\right].\end{aligned}$$

3. Let (X_0', X_1') denote the endpoints of the learned ODE. By the Cauchy–Schwarz inequality,

$$\|X_1' - X_0'\|^2 \leq \int_0^1 \|v^*(X_t', t)\|^2 \, dt.$$

4. Since X_t' and X_t have the same marginal distribution,

$$\mathbb{E}\|X_1' - X_0'\|^2 \leq \mathbb{E}\|X_1 - X_0\|^2.$$

5. Thus, exact rectification cannot increase the expected squared endpoint displacement.

Rectified flow versus optimal transport

1. Both approaches seek efficient transport between the same endpoint distributions.
2. Optimal transport minimizes a global cost:

$$W_2^2(p_0, p_1) = \min_{\pi \in \Pi(p_0, p_1)} \mathbb{E}_\pi \|X_1 - X_0\|^2.$$

3. Rectified flow learns a velocity field from interpolated endpoint pairs and can iteratively improve their coupling through reflow.
4. Important distinctions:
   - Straight interpolation does not guarantee straight ODE trajectories.
   - Reduced transport cost does not guarantee a globally optimal coupling.
   - Straight trajectories alone do not establish optimal transport.
5. Practical motivation: straighter trajectories can often be approximated accurately using fewer numerical integration steps.

Why flow matching on manifolds?

1. Standard flow matching assumes $X_t \in \mathbb{R}^d$ and learns $v_t(x) \in \mathbb{R}^d$.
2. Many data objects obey geometric constraints:
   - Unit directions: $S^{d-1} = \{x : \|x\| = 1\}$.
   - Rotations: $SO(3) = \{R : R^\top R = I, \det R = 1\}$.
   - Periodic angles: the torus $\mathbb{T}^d$.
3. Straight interpolation $(1-t)x_0 + tx_1$ can leave the valid state space.
4. Goal: learn probability transport while keeping generated trajectories on the manifold.

What is a manifold?

1. A smooth d-dimensional manifold $\mathcal{M}$ locally resembles $\mathbb{R}^d$, even if it is curved globally.
2. Examples: $S^2 \subset \mathbb{R}^3$ has intrinsic dimension 2; $SO(3)$ has intrinsic dimension 3.
3. A point $x \in \mathcal{M}$ has a tangent space $T_x\mathcal{M}$ of admissible infinitesimal directions.
4. Unlike $\mathbb{R}^d$, different points generally have different tangent spaces.
5. A vector field assigns $v_t(x) \in T_x\mathcal{M}$ to each point and time.

Tangent spaces on the sphere

1. For $S^{d-1} = \{x \in \mathbb{R}^d : \|x\| = 1\}$, differentiate $x^\top x = 1$ along a curve:

$$2x^\top \dot{x} = 0.$$

2. Thus the tangent space is

$$\boxed{T_x S^{d-1} = \{u \in \mathbb{R}^d : x^\top u = 0\}.}$$

3. An ambient vector $a \in \mathbb{R}^d$ can be projected onto this tangent space:

$$\boxed{P_x a = (I - xx^\top)a.}$$

4. A tangent velocity satisfies $x^\top v_t(x) = 0$, preserving the unit-norm constraint in exact continuous dynamics.

Riemannian metrics and norms

1. A Riemannian metric $g_x(u, w)$ is an inner product on $T_x\mathcal{M}$ that varies smoothly with x.
2. It defines tangent-vector lengths:

$$\|u\|_{g_x}^2 = g_x(u, u).$$

3. On a sphere with its standard metric, $g_x(u, w) = u^\top w$ for tangent vectors.
4. The metric defines lengths of curves, geodesic distance, volume, gradient, and divergence.
5. In manifold flow matching, the squared prediction error must compare vectors in the same tangent space.

Geodesics replace straight lines

1. In Euclidean space, the shortest path between x_0 and x_1 is a straight line.
2. On a manifold, a geodesic is a curve with zero covariant acceleration; locally it is length-minimizing.
3. A minimizing geodesic need not be unique (for example, antipodal points on a sphere).
4. The exponential map $\mathrm{Exp}_x(u)$ follows the geodesic starting at x with initial tangent velocity u for unit time.
5. Locally, the logarithm map $\mathrm{Log}_x(y)$ returns a tangent vector whose exponential reaches y.

Geodesic interpolation

1. For endpoints connected by a selected minimizing geodesic, define

$$\boxed{X_t = \operatorname{Exp}_{X_0}\left(t \operatorname{Log}_{X_0}(X_1)\right), \quad t \in [0,1].}$$

2. At $t = 0$, $X_t = X_0$; at $t = 1$, $X_t = X_1$.
3. Its velocity $\dot{X}_t \in T_{X_t}\mathcal{M}$ is tangent along the entire curve.
4. In $\mathbb{R}^d$, $\operatorname{Exp}_x(u) = x + u$ and $\operatorname{Log}_x(y) = y - x$.
5. Therefore geodesic interpolation reduces to $(1-t)X_0 + tX_1$ in Euclidean space.

Example: great-circle interpolation on a sphere

1. Let $x_0, x_1 \in S^{d-1}$ with $x_0 \neq -x_1$ and angle $\omega = \arccos(x_0^\top x_1) \in (0, \pi)$.
2. The minimizing great-circle interpolation is

$$\boxed{x_t = \frac{\sin((1-t)\omega)}{\sin \omega} x_0 + \frac{\sin(t\omega)}{\sin \omega} x_1.}$$

3. The curve stays on the sphere: $\|x_t\| = 1$.
4. Its speed is constant and equals ω under the standard metric.
5. As $\omega \to 0$, the path becomes constant; antipodal endpoints require choosing a great circle.

Exponential and logarithm maps on the sphere

1. For $u \in T_x S^{d-1}$ and $r = \|u\|$, the exponential map is

$$\mathrm{Exp}_x(u) = \cos(r)x + \frac{\sin r}{r}u.$$

2. For $y \neq -x$, let $\omega = \arccos(x^\top y)$. Then

$$\mathrm{Log}_x(y) = \frac{\omega}{\sin \omega}(y - \cos \omega\, x).$$

3. Both formulas use continuous limits at $r = 0$ and $\omega = 0$.
4. Substitution into geodesic interpolation recovers the great-circle formula.

Probability distributions on manifolds

1. Let $d\,\mathrm{vol}_g(x)$ be the Riemannian volume measure on $\mathcal{M}$.
2. A density $p_t(x)$ is defined relative to this measure:

$$\mathbb{P}(X_t \in A) = \int_A p_t(x)\, d\,\mathrm{vol}_g(x).$$

3. Normalization requires

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

4. A probability path $\{p_t\}_{t\in[0,1]}$ connects a tractable p_0 to $p_1 = p_{\mathrm{data}}$.
5. A manifold density is not generally a density with respect to ambient Lebesgue measure in $\mathbb{R}^D$.

Riemannian gradient and divergence

1. The Riemannian gradient $\operatorname{grad}_{\mathcal{M}} f$ is characterized by

$$g_x(\operatorname{grad}_{\mathcal{M}} f, u) = df_x[u], \quad u \in T_x\mathcal{M}.$$

2. The divergence $\operatorname{div}_{\mathcal{M}} v$ measures local net outward flux relative to Riemannian volume.
3. In local coordinates with metric matrix $G = (g_{ij})$,

$$\operatorname{div}_{\mathcal{M}} v = \frac{1}{\sqrt{|G|}} \sum_i \partial_i \left(\sqrt{|G|}\, v^i \right).$$

4. In Euclidean coordinates, $G = I$ and this becomes $\nabla \cdot v$.

The Riemannian continuity equation

1. Probability is conserved under a tangent velocity field v_t:

$$\boxed{\partial_t p_t + \operatorname{div}_{\mathcal{M}}(p_t v_t) = 0.}$$

2. For any smooth region $A \subset \mathcal{M}$ with outward boundary normal n,

$$\frac{d}{dt} \int_A p_t \, d\operatorname{vol}_g = - \int_{\partial A} p_t g(v_t, n) \, dS_g.$$

3. The left side is the rate of change of probability in A; the right side is negative outward probability flux.
4. This is the geometric analogue of the Euclidean continuity equation.

Why the continuity equation follows from transport

1. Let X_t follow the manifold ODE $\dot{X}_t = v_t(X_t)$.
2. For a smooth test function φ,

$$\frac{d}{dt}\mathbb{E}[\varphi(X_t)] = \mathbb{E}[d\varphi_{X_t}[v_t(X_t)]].$$

3. Writing the expectation as an integral gives

$$\frac{d}{dt}\int_{\mathcal{M}} \varphi p_t \, d\,\mathrm{vol}_g = \int_{\mathcal{M}} g(\mathrm{grad}\,\varphi, v_t) p_t \, d\,\mathrm{vol}_g \,.$$

4. Integration by parts (with vanishing boundary flux) yields

$$\int_{\mathcal{M}} \varphi\big[\partial_t p_t + \mathrm{div}_{\mathcal{M}}(p_t v_t)\big] d\,\mathrm{vol}_g = 0.$$

5. Since this holds for all test functions, the continuity equation follows.

Conditional paths and endpoint couplings

1. Choose a coupling $\pi(X_0, X_1)$ with marginals p_0 and $p_1 = p_{\text{data}}$.
2. For each sampled endpoint pair, construct a differentiable manifold-valued path $X_t = \gamma_t(X_0, X_1)$.
3. The path has conditional target velocity

$$u_t(X_t \mid X_0, X_1) = \partial_t \gamma_t(X_0, X_1) \in T_{X_t}\mathcal{M}.$$

4. Geodesic interpolation is a natural choice, but not the only admissible path.
5. Averaging over sampled endpoint pairs defines the marginal probability path p_t.

Riemannian conditional flow matching

1. Parameterize a tangent vector field $v_\theta(x, t) \in T_x\mathcal{M}$.
2. Sample $(X_0, X_1) \sim \pi$, $t \sim \mathcal{U}[0, 1]$, and set $X_t = \gamma_t(X_0, X_1)$.
3. Regress onto the conditional tangent velocity:

$$\boxed{\mathcal{L}_{\mathrm{CFM}}(\theta) = \mathbb{E}\left[\left\| v_\theta(X_t, t) - \dot{X}_t \right\|_{g_{X_t}}^2\right].}$$

4. The prediction and target both lie in $T_{X_t}\mathcal{M}$; no cross-tangent-space subtraction is needed.
5. As in Euclidean flow matching, training does not require solving the generative ODE.

Why the optimal predictor is a conditional mean

1. At fixed $(X_t = x, t)$, the target $U = \dot{X}_t$ is a random vector in the same inner-product space $T_x\mathcal{M}$.
2. For any candidate $a \in T_x\mathcal{M}$, squared-error decomposition gives

$$\mathbb{E}[\|U - a\|_{g_x}^2 \mid X_t = x] = \mathbb{E}[\|U - m\|_{g_x}^2 \mid X_t = x] + \|m - a\|_{g_x}^2,$$

   where $m = \mathbb{E}[U \mid X_t = x]$.
3. Therefore the optimal marginal velocity is

$$\boxed{v_t^*(x) = \mathbb{E}[\dot{X}_t \mid X_t = x].}$$

4. This is an ordinary conditional expectation inside the fixed tangent space $T_x\mathcal{M}$.

Why the marginal velocity transports the path

1. For a smooth test function φ, the conditional path satisfies
$$\frac{d}{dt}\mathbb{E}[\varphi(X_t)] = \mathbb{E}[d\varphi_{X_t}[\dot{X}_t]].$$
2. Condition on X_t and use linearity of the differential:
$$= \mathbb{E}\Big[d\varphi_{X_t}\Big[\mathbb{E}[\dot{X}_t \mid X_t]\Big]\Big] .$$
3. Hence
$$\frac{d}{dt}\int \varphi p_t \, d\,\mathrm{vol}_g = \int g(\mathrm{grad}\,\varphi, v_t^*) p_t \, d\,\mathrm{vol}_g .$$
4. Integration by parts gives $\partial_t p_t + \mathrm{div}_{\mathcal{M}}(p_t v_t^*) = 0$.
5. Thus the learned marginal velocity reproduces the prescribed marginal path under suitable regularity.

A simple example on the circle

1. The circle is $S^1 = \{(\cos\phi, \sin\phi) : \phi \in \mathbb{R}/(2\pi\mathbb{Z})\}$.
2. For angles ϕ_0, ϕ_1, choose the shortest signed difference

$$\delta = \mathrm{wrap}_{(-\pi,\pi]}(\phi_1 - \phi_0).$$

3. A geodesic interpolation is

$$\phi_t = \phi_0 + t\delta \pmod{2\pi}, \qquad x_t = (\cos\phi_t, \sin\phi_t).$$

4. Its ambient tangent velocity is

$$\dot{x}_t = \delta(-\sin\phi_t, \cos\phi_t).$$

5. At antipodal points ($|\delta| = \pi$), the shortest direction is ambiguous and a branch must be chosen.

Rotations form the manifold SO(3)

1. Three-dimensional rotations are matrices $R \in SO(3)$ satisfying $R^\top R = I$ and $\det R = 1$.
2. Their tangent vectors have the form

$$\boxed{T_R SO(3) = \{R\Omega : \Omega^\top = -\Omega\}.}$$

3. The skew-symmetric matrix Ω represents an angular velocity.
4. A valid rotation trajectory satisfies

$$\dot{R}_t = R_t \Omega_t, \qquad \Omega_t^\top = -\Omega_t.$$

5. Naively interpolating matrix entries generally violates the orthogonality constraint.

Geodesic interpolation on SO(3)

1. Under a standard bi-invariant metric, a selected shortest geodesic from R_0 to R_1 is

$$\boxed{R_t = R_0 \exp\Big(t \log(R_0^\top R_1)\Big)\,.}$$

2. Here exp is the matrix exponential and log is a suitable skew-symmetric matrix logarithm.
3. Differentiation gives the tangent velocity

$$\dot{R}_t = R_t \Omega, \qquad \Omega = \log(R_0^\top R_1).$$

4. Rotations of angle π may have nonunique shortest geodesics; the logarithm branch must be handled carefully.
5. This is useful for rigid-body orientations and molecular or protein configurations.

Constructing tangent neural vector fields

1. A neural network may output an ambient vector $a_\theta(x, t)$, which need not be tangent.
2. On the sphere, project the prediction:

$$\boxed{v_\theta(x, t) = (I - xx^\top)a_\theta(x, t).}$$

3. For $SO(3)$, predict a skew-symmetric $\Omega_\theta(R, t)$ and use

$$v_\theta(R, t) = R\Omega_\theta(R, t).$$

4. Intrinsic-coordinate and equivariant architectures offer alternatives.
5. Tangency enforces the infinitesimal constraint; numerical integration must also respect manifold geometry.

Sampling with manifold-aware integration

1. Begin with $X_0 \sim p_0$ on $\mathcal{M}$ and solve

$$\dot{X}_t = v_\theta(X_t, t), \qquad t : 0 \to 1.$$

2. A first-order manifold Euler update is

$$\boxed{x_{k+1} = \mathrm{Exp}_{x_k}(h v_\theta(x_k, t_k))\,.}$$

3. On S^{d-1}, one can use the spherical exponential map or a suitable retraction.
4. On $SO(3)$, an update is $R_{k+1} = R_k \exp(h\Omega_\theta(R_k, t_k))$.
5. Unlike ambient Euler addition, these updates preserve the manifold constraint up to numerical error in the map evaluation.

Retractions and numerical tradeoffs

1. Computing exact exponential maps may be expensive on some manifolds.
2. A retraction $\mathrm{Retr}_x(u)$ approximates $\mathrm{Exp}_x(u)$ locally and satisfies

$$\mathrm{Retr}_x(0) = x, \qquad D\,\mathrm{Retr}_x(0) = \mathrm{id}_{T_x\mathcal{M}}\,.$$

3. On the sphere, a simple retraction is

$$\mathrm{Retr}_x(u) = \frac{x+u}{\|x+u\|}, \qquad u \in T_x S^{d-1}.$$

4. Retractions preserve the constraint but introduce numerical approximation error.
5. Higher-order manifold integrators can improve accuracy when sampling uses few steps.

Laplace–Beltrami operator and manifold diffusion

1. Recall the Euclidean Laplacian: $\Delta f = \nabla \cdot (\nabla f)$.
2. On a Riemannian manifold, its analogue is

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

3. For isotropic Brownian diffusion with constant diffusivity D,

$$\partial_t p_t = D \Delta_{\mathcal{M}} p_t.$$

4. With drift $b_t \in T\mathcal{M}$, the manifold Fokker–Planck equation is

$$\partial_t p_t = -\operatorname{div}_{\mathcal{M}}(p_t b_t) + D \Delta_{\mathcal{M}} p_t.$$

5. Thus the earlier gradient–divergence–Laplacian discussion extends directly to curved spaces.

Probability flow velocity on a manifold

1. For constant isotropic diffusivity D, rewrite the manifold Fokker–Planck equation as

$$\partial_t p_t = -\operatorname{div}_{\mathcal{M}}(p_t b_t - D \operatorname{grad}_{\mathcal{M}} p_t)\,.$$

2. Wherever $p_t > 0$, use $\operatorname{grad}_{\mathcal{M}} p_t = p_t \operatorname{grad}_{\mathcal{M}} \log p_t$.
3. Therefore the same density evolution is generated by the deterministic velocity

$$\boxed{v_t = b_t - D \operatorname{grad}_{\mathcal{M}} \log p_t.}$$

4. This is the manifold counterpart of the probability flow ODE; it matches one-time marginals, not stochastic sample paths.

Manifold flow matching and optimal transport

1. In Euclidean optimal transport, a common cost is $\|x_1 - x_0\|^2$.
2. On a Riemannian manifold, the analogous intrinsic cost is $d_{\mathcal{M}}(x_0, x_1)^2$.
3. An endpoint coupling can minimize

$$\inf_{\pi \in \Pi(p_0, p_1)} \mathbb{E}_{\pi}[d_{\mathcal{M}}(X_0, X_1)^2].$$

4. Geodesic interpolation provides short conditional paths for the chosen coupling.
5. However, using geodesics with an arbitrary coupling does not automatically produce a globally optimal transport map.

Applications and practical limitations

1. Applications include molecular orientations, rigid-body poses, directional data, periodic variables, and constrained scientific states.
2. Practical choices matter:
   - Source distribution p_0 on the manifold (uniform, concentrated, or learned).
   - Endpoint coupling and interpolation path.
   - Handling cut loci, nonunique geodesics, and coordinate singularities.
   - Tangent parameterization and stable numerical integration.
3. Some constrained data spaces are not smooth manifolds everywhere; additional care or different models may be needed.

Continuous, manifold, and discrete flow matching

1. Euclidean flow matching: $X_t \in \mathbb{R}^d$; learn $v_t(x)$ and solve an ODE.
2. Manifold flow matching: $X_t \in \mathcal{M}$; learn $v_t(x) \in T_x\mathcal{M}$ and solve a geometric ODE.
3. Discrete flow matching: X_t belongs to a finite or countable state space; model probability transfer through jump rates.
4. Probability evolution is expressed by

$$\begin{aligned} \mathbb{R}^d: &\quad \partial_t p_t = -\nabla \cdot (p_t v_t), \\ \mathcal{M}: &\quad \partial_t p_t = -\operatorname{div}_{\mathcal{M}}(p_t v_t), \\ \text{Discrete}: &\quad \dot{p}_t(y) = \sum_x p_t(x) R_t(x, y). \end{aligned}$$

5. All three express conservation of probability along a prescribed path.

Summary of Riemannian flow matching

1. Replace Euclidean state space by a smooth manifold $\mathcal{M}$ and Euclidean velocity by tangent velocity.
2. Replace straight interpolation by valid manifold paths, often geodesics.
3. Conditional regression still learns a marginal velocity through conditional expectation in $T_x\mathcal{M}$.
4. Probability conservation becomes

$$\partial_t p_t + \operatorname{div}_{\mathcal{M}}(p_t v_t) = 0.$$

5. Generate by integrating a learned tangent vector field using manifold-aware numerical methods.
6. The central principle of flow matching survives; the geometry changes how paths, velocities, and differential operators are defined.

Two independent notions of discreteness

1. Time and state space are independent modeling choices:
   - Discrete time: $t = 0, 1, \ldots, T$; continuous time: $t \in [0, T]$.
   - Continuous state: $z_t \in \mathbb{R}^d$; discrete state: $z_t \in \mathcal{V}^L$.
2. Four combinations arise:

Time	State	Example
Discrete	Continuous	Gaussian DDPM
Continuous	Continuous	Diffusion SDE, flow ODE
Discrete	Discrete	Categorical diffusion
Continuous	Discrete	Continuous-time Markov chain

3. Deterministic versus stochastic dynamics is a third distinction.
4. A continuous-time model may be simulated using discrete numerical steps.

Why discrete flow matching?

1. Existing flow matching learns a velocity field for $X_t \in \mathbb{R}^d$:

$$\frac{dx_t}{dt} = v_t(x_t), \qquad \partial_t p_t = -\nabla \cdot (p_t v_t).$$

2. Tokens, molecular identities, and categorical features live in a finite state space $\mathcal{S}$; a token cannot move infinitesimally between categories.
3. Discrete flow matching (DFM) instead prescribes a path of categorical distributions and constructs stochastic jump dynamics realizing that path.
4. Time remains continuous, $t \in [0, 1]$, even though states are discrete.

Continuous and discrete flow matching: common structure

1. Both start from a simple source p_0 and seek a target $p_1 = p_{\text{data}}$.
2. Both choose intermediate marginals $\{p_t\}_{t\in[0,1]}$ and learn dynamics consistent with them.
3. Continuous case: probability moves through a velocity field and obeys a continuity equation.
4. Discrete case: probability moves along transitions between states and obeys a master equation.
5. The analogue of a velocity field is a transition-rate matrix (or a probability-velocity representation used to construct one).

Discrete states and continuous time

1. For one token, $\mathcal{S} = \{1, \ldots, K\}$; for length-$L$ sequences, $\mathcal{S} = \mathcal{V}^L$.
2. A categorical distribution assigns probabilities $p_t(i) \geq 0$ with $\sum_i p_t(i) = 1$.
3. A continuous-time Markov chain (CTMC) stays in a state for a random duration and then jumps to another state.
4. For $i \neq j$, $R_t(i,j) \geq 0$ is the instantaneous jump rate $i \to j$:

$$\Pr(X_{t+h} = j \mid X_t = i) = hR_t(i,j) + o(h).$$

5. Set $R_t(i,i) = -\sum_{j \neq i} R_t(i,j)$, so every row sums to zero.

The master equation: discrete probability conservation

1. The distribution of a CTMC evolves according to

$$\boxed{\frac{d}{dt}p_t(j) = \sum_i p_t(i)R_t(i,j).}$$

2. Equivalently, separate incoming and outgoing probability flux:

$$\dot{p}_t(j) = \sum_{i \neq j} \underbrace{p_t(i)R_t(i,j)}_{\text{inflow from } i} - \sum_{i \neq j} \underbrace{p_t(j)R_t(j,i)}_{\text{outflow to } i}.$$

3. Summing over j gives $\frac{d}{dt}\sum_j p_t(j) = 0$ because the rate-matrix rows sum to zero.
4. This is the discrete analogue of $\partial_t p_t + \nabla \cdot (p_t v_t) = 0$.

Deriving the master equation from a small time step

1. For small $h > 0$, conditional transition probabilities satisfy

$$\Pr(X_{t+h} = j \mid X_t = i) = \delta_{ij} + hR_t(i,j) + o(h).$$

2. Apply the law of total probability:

$$p_{t+h}(j) = \sum_i p_t(i)\big[\delta_{ij} + hR_t(i,j) + o(h)\big].$$

3. Subtract $p_t(j)$, divide by h, and take $h \to 0$:

$$\boxed{\dot{p}_t(j) = \sum_i p_t(i) R_t(i,j).}$$

4. Unlike a deterministic ODE, the generator describes random jumps, not a derivative of the token label.

Probability flux on a graph

1. Regard categorical states as vertices of a directed graph; allowed transitions are edges.
2. Define directed probability flux $J_t(i,j) = p_t(i)R_t(i,j)$ for $i \neq j$.
3. The net antisymmetric flux between states is

$$F_t(i,j) = J_t(i,j) - J_t(j,i) = -F_t(j,i).$$

4. The master equation becomes

$$\boxed{\dot{p}_t(j) = \sum_{i \neq j} F_t(i,j).}$$

5. Just as divergence measures net outflow in a continuous fluid, the net flux across graph edges determines changes in probability at each state.

Conditional and marginal discrete probability paths

1. Draw a target sample $X_1 \sim p_{\text{data}}$ and specify a tractable conditional path $p_t(x \mid x_1)$.
2. Average over targets to obtain the desired marginal path:

$$\boxed{p_t(x) = \sum_{x_1 \in \mathcal{S}} p_t(x \mid x_1) p_{\text{data}}(x_1).}$$

3. Require $p_0(x \mid x_1) = p_0(x)$ and $p_1(x \mid x_1) = 1[x = x_1]$.
4. Therefore $p_{t=0} = p_0$ and $p_{t=1} = p_{\text{data}}$.
5. As in conditional flow matching, conditioning on known training samples makes the path easy to sample without evaluating p_{data}.

A simple mixture probability path

1. Let κ_t increase from $\kappa_0 = 0$ to $\kappa_1 = 1$.
2. For one categorical variable, choose

$$\boxed{p_t(x \mid x_1) = (1 - \kappa_t)p_0(x) + \kappa_t 1[x = x_1].}$$

3. Sampling is simple: with probability $1 - \kappa_t$ draw from p_0; otherwise use x_1.
4. At $t = 0$ we obtain source noise; at $t = 1$ the conditional distribution collapses to the clean target.
5. Interpolation occurs in *probability space*, not by taking numerical averages of token IDs.

Masked paths for language generation

1. Introduce a special mask token m and let $p_0 = \delta_m$ for one position.
2. A conditional path from mask to clean token a is

$$\boxed{p_t(x \mid a) = (1 - \kappa_t)1[x = m] + \kappa_t 1[x = a].}$$

3. For a length-L sequence x_1, a common construction is

$$p_t(x \mid x_1) = \prod_{\ell=1}^{L} \left[(1 - \kappa_t)1[x^\ell = m] + \kappa_t 1[x^\ell = x_1^\ell]\right].$$

4. Conditional independence of positions does not imply that the marginal distribution of sequences factorizes: the clean-data distribution couples positions.

Example: a three-token masked path

1. Consider a clean sequence $x_1 = (A, B, C)$ and $\kappa_t = t$.
2. Each position independently equals its clean token with probability t, and mask m with probability $1 - t$.
3. At $t = \frac{1}{2}$, for example,

$$\Pr(X_t = (A, m, C) \mid X_1 = (A, B, C)) = \left(\tfrac{1}{2}\right)^3 = \tfrac{1}{8}.$$

4. The expected number of revealed tokens is Lt (here $3t$).
5. At $t = 0$: (m, m, m); at $t = 1$: (A, B, C) with probability one.

Constructing conditional transition rates

1. For the masked single-token path, the state is either m or the target token a.
2. Let $R_t(m, a \mid a) = \lambda_t$ and let a be absorbing; all other off-diagonal rates are zero.
3. The master equation for the masked probability is

$$\frac{d}{dt} p_t(m \mid a) = -\lambda_t p_t(m \mid a).$$

4. Since $p_t(m \mid a) = 1 - \kappa_t$, matching derivatives gives

$$\boxed{\lambda_t = \frac{\dot{\kappa}_t}{1 - \kappa_t}.}$$

5. The rate may diverge as $t \to 1$; this enforces complete unmasking by the endpoint.

Why the conditional rates generate the path

1. The only nonzero transition is $m \to a$ with rate $\lambda_t = \dot{\kappa}_t/(1-\kappa_t)$.
2. For the mask state,

$$\dot{p}_t(m \mid a) = -\lambda_t(1-\kappa_t) = -\dot{\kappa}_t.$$

3. For the target state,

$$\dot{p}_t(a \mid a) = +\lambda_t(1-\kappa_t) = +\dot{\kappa}_t.$$

4. These agree exactly with derivatives of

$$p_t(m \mid a) = 1-\kappa_t, \qquad p_t(a \mid a) = \kappa_t.$$

5. This is the discrete analogue of verifying that a conditional velocity field satisfies the continuity equation.

The posterior over clean data

1. The marginal generator depends on the unknown clean target X_1 given the current state $X_t = x$.
2. Bayes' rule defines the clean-data posterior:

$$\boxed{p(X_1 = a \mid X_t = x) = \frac{p_t(x \mid x_1 = a)p_{\text{data}}(a)}{p_t(x)}.}$$

3. In language modeling, this is a distribution over possible clean tokens or clean sequences conditioned on a partially corrupted sequence.
4. A neural network $q_\theta(a \mid x, t)$ approximates this posterior; it can use context from all token positions.
5. We never need to evaluate $p_{\text{data}}(a)$ explicitly to train the posterior predictor.

From conditional rates to marginal rates

1. Suppose each conditional path is generated by rates $R_t(x, y \mid a)$.
2. Define the marginal rate matrix through posterior averaging:

$$\boxed{R_t(x, y) = \mathbb{E}[R_t(x, y \mid X_1) \mid X_t = x], \quad y \neq x.}$$

3. Why does this work? Multiply by $p_t(x)$:

$$p_t(x) R_t(x, y) = \sum_a p_{\text{data}}(a) p_t(x \mid a) R_t(x, y \mid a).$$

4. Summing the conditional master equations over a yields the marginal master equation for p_t.
5. In practice replace the exact posterior with q_θ to obtain learned rates R_t^θ.

A direct denoiser-to-rate formula for masking

1. For a single masked token, conditional rates satisfy

$$R_t(m, a \mid X_1) = \frac{\dot{\kappa}_t}{1 - \kappa_t} 1[X_1 = a], \qquad a \neq m.$$

2. Averaging over the posterior gives

$$\boxed{R_t^\theta(m, a) = \frac{\dot{\kappa}_t}{1 - \kappa_t} q_\theta(a \mid m, t).}$$

3. For sequences, q_θ predicts clean tokens using the entire current sequence; the rate at each masked position is conditioned on that context.
4. In the simple absorbing-mask construction, already revealed positions are not remasked.
5. The network predicts *where to jump*; the schedule controls *how quickly jumps occur*.

Training discrete flow matching

1. Draw $t \sim \mathcal{U}[0,1]$, $X_1 \sim p_{\text{data}}$, and $X_t \sim p_t(\cdot \mid X_1)$.
2. Train a denoiser to predict clean data from the intermediate state:

$$\boxed{\mathcal{L}_{\text{DFM}}(\theta) = \mathbb{E}_{t,X_1,X_t}[-\log q_\theta(X_1 \mid X_t, t)].}$$

3. For token-wise prediction, a common loss is

$$\mathcal{L}_{\text{DFM}}(\theta) = \mathbb{E}\left[-\sum_{\ell=1}^{L} \log q_\theta(X_1^\ell \mid X_t, t, \ell)\right].$$

4. Under sufficient model capacity, cross-entropy minimization recovers the appropriate conditional posterior (or its token marginals).
5. This is simulation-free training: no CTMC trajectory needs to be simulated during optimization.

Why cross-entropy learns the posterior

1. Fix an intermediate state $X_t = x$ and time t.
2. Conditional expected loss is

$$\mathcal{L}_{x,t}(q) = -\sum_a p(X_1 = a \mid X_t = x) \log q(a \mid x, t).$$

3. Add and subtract the entropy of the true posterior:

$$\mathcal{L}_{x,t}(q) = H(p(X_1 = \cdot \mid X_t = x)) + D_{\mathrm{KL}}(p(X_1 = \cdot \mid X_t = x) \| q(\cdot \mid x, t)) .$$

4. The KL term is nonnegative and vanishes exactly when $q = p(X_1 = \cdot \mid X_t = x)$ (on the posterior support).
5. Thus, the training objective estimates the posterior required to construct marginal jump rates.

Sampling: Euler approximation of a CTMC

1. Start with $X_0 \sim p_0$ and evaluate learned rates $R_t^\theta(X_t, y)$.
2. For a small step h, define

$$\Pr(X_{t+h} = y \mid X_t = x) \approx \begin{cases} hR_t^\theta(x, y), & y \neq x, \\ 1 - h\sum_{y \neq x} R_t^\theta(x, y), & y = x. \end{cases}$$

3. Sample the next state from these probabilities and repeat until near $t = 1$.
4. Choose h so the stay probability is nonnegative; near singular endpoints use adaptive steps, exact jump-time simulation, or a final denoising step.
5. For sequences, sparse single-position jumps avoid enumerating the full state space $|\mathcal{V}|^L$.

Sampling: exact jump times for a masked token

1. The mask survives from time 0 to t with probability

$$\Pr(T_{\text{jump}} > t) = \exp\left(-\int_0^t \lambda_s \, ds\right) = 1 - \kappa_t.$$

2. Therefore the first jump time can be sampled by drawing $U \sim \mathcal{U}(0,1)$ and solving

$$\boxed{\kappa_{T_{\text{jump}}} = U.}$$

3. At the jump time, sample a clean token using the learned posterior conditioned on the current context.
4. For multiple positions, posterior predictions can change after other tokens are revealed, so an event-driven sampler must update relevant rates.
5. Exact event times are possible for the simple hazard; learned context-dependent jump destinations still require model evaluations.

Choice of scheduler and path

1. The schedule κ_t determines the intermediate corruption level and the jump hazard
$$\lambda_t = \frac{\dot{\kappa}_t}{1 - \kappa_t}.$$
2. For $\kappa_t = t$, the hazard is $\lambda_t = (1-t)^{-1}$.
3. A nonlinear schedule changes when information is revealed, even when source and target distributions are unchanged.
4. Beyond masking, paths can use uniform token replacement, endpoint mixtures, or domain-specific transitions.
5. The same marginal path may admit multiple rate matrices: choosing a path does not uniquely specify individual jump trajectories.

General endpoints and couplings

1. Continuous flow matching can choose a coupling $\pi(x_0, x_1)$; discrete flow matching has the same freedom.
2. For sampled endpoints x_0, x_1, a coordinate-wise path is

$$p_t(x^\ell \mid x_0, x_1) = (1 - \kappa_t)1[x^\ell = x_0^\ell] + \kappa_t 1[x^\ell = x_1^\ell].$$

3. The sequence conditional path can factorize across positions given endpoints.
4. Average over the endpoint coupling:

$$p_t(x) = \sum_{x_0, x_1} p_t(x \mid x_0, x_1)\pi(x_0, x_1).$$

5. Different couplings and schedules can change intermediate distributions and sampling efficiency.

What is and is not unique?

1. A probability path p_t specifies how much probability each state carries at each time.
2. A rate matrix R_t specifies the stochastic transitions realizing that evolution.
3. The master equation constrains only net flux at each state, so multiple generators can realize the same path.
4. This parallels continuous flow matching: a probability path need not determine a unique velocity field.
5. Additional principles, such as transition sparsity, kinetic cost, or domain constraints, can guide the choice of generator.

Discrete flow matching versus discrete diffusion

1. Both can use categorical corruption paths, denoising objectives, and continuous-time Markov chains.
2. Discrete diffusion is often presented as a prescribed forward corruption process followed by learned reverse transitions.
3. Discrete flow matching starts from a desired source-to-data probability path and constructs generating jump rates, often using a learned clean-data posterior.
4. These are overlapping frameworks, not mutually exclusive model classes; some CTMC diffusion constructions can be expressed as discrete flows.
5. The key difference is the *construction and parameterization* of the dynamics, not simply whether the states or time are discrete.

Continuous versus discrete flow matching: summary

1. *Continuous states:* $\frac{dx_t}{dt} = v_t(x_t)$ and $\partial_t p_t = -\nabla \cdot (p_t v_t)$.
2. *Discrete states:* CTMC rates $R_t(x, y)$ and $\dot{p}_t(y) = \sum_x p_t(x) R_t(x, y)$.
3. *Conditional training:* regress a conditional velocity in the continuous case; predict clean-token posteriors (often via cross-entropy) in the discrete case.
4. *Generation:* integrate an ODE versus simulate stochastic jumps.
5. *Shared principle:* choose tractable conditional probability paths, average to the marginal path, and learn dynamics that reproduce it.

Further directions and applications

1. *Language:* iterative, potentially parallel token generation rather than strictly left-to-right autoregression.
2. *Molecules and proteins:* jointly generate discrete identities (atom or residue types) and continuous coordinates using coupled jump and ODE dynamics.
3. *Structured discrete data:* constrain allowed transitions to respect validity, graph structure, or physical rules.
4. *Open design choices:* probability paths, endpoint couplings, transition graphs, rate parameterizations, and efficient samplers.
5. DFM is a generalization of the probability-transport viewpoint, not merely an ODE with token IDs substituted for real coordinates.

Outline

1 Introduction

2 Flow Matching

3 Diffusion Models
- Forward Diffusion Process
- Reverse Diffusion Process
- Training and Sampling
- Predicting the Noise
- Discrete Diffusion and Language Models

4 Score Matching

5 Connections

6 Frontiers

7 Latent Variable Models

8 Variational Inference

Review: multivariate Gaussian distribution

1. A D-dimensional random vector $\mathrm{x} \in \mathbb{R}^D$ follows a multivariate Gaussian distribution if its probability density is

$$\boxed{\mathcal{N}(\mathrm{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathrm{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathrm{x}-\boldsymbol{\mu})\right\}}$$

   where $\boldsymbol{\mu}$ is the mean vector and $\boldsymbol{\Sigma}$ is the covariance matrix.

2. For an isotropic Gaussian, $\boldsymbol{\Sigma} = \sigma^2 \mathrm{I}$, giving

$$\mathcal{N}(\mathrm{x} \mid \boldsymbol{\mu}, \sigma^2 \mathrm{I}) = \frac{1}{(2\pi\sigma^2)^{D/2}} \exp\left\{-\frac{\|\mathrm{x}-\boldsymbol{\mu}\|^2}{2\sigma^2}\right\}.$$

Review: linear transformations and sums of Gaussians

1. If
$$\mathsf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}),$$
then an affine transformation remains Gaussian:
$$\boxed{\mathsf{A}\mathsf{x} + \mathsf{b} \sim \mathcal{N}(\mathsf{A}\boldsymbol{\mu} + \mathsf{b}, \mathsf{A}\boldsymbol{\Sigma}\mathsf{A}^T).}$$
2. For two *independent* Gaussian random vectors,
$$\mathsf{x} \sim \mathcal{N}(\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x), \qquad \mathsf{y} \sim \mathcal{N}(\boldsymbol{\mu}_y, \boldsymbol{\Sigma}_y),$$
their sum is also Gaussian:
$$\boxed{\mathsf{x} + \mathsf{y} \sim \mathcal{N}(\boldsymbol{\mu}_x + \boldsymbol{\mu}_y, \boldsymbol{\Sigma}_x + \boldsymbol{\Sigma}_y).}$$

Review: sampling and reparameterization

1. A Gaussian random vector

$$\mathrm{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

   can be generated as

$$\boxed{\mathrm{x} = \boldsymbol{\mu} + \mathrm{L}\boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathrm{I})}$$

   where $\mathrm{L}\mathrm{L}^T = \boldsymbol{\Sigma}$.

2. For an isotropic Gaussian, this simplifies to

$$\boxed{\mathrm{x} = \boldsymbol{\mu} + \sigma\boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathrm{I}).}$$

Sampling and reparameterization: proof

1. Suppose

$$\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathsf{I})$$

and define

$$\boxed{\mathsf{x} = \boldsymbol{\mu} + \mathsf{L}\boldsymbol{\epsilon}}$$

where $\mathsf{L}\mathsf{L}^T = \boldsymbol{\Sigma}$.

2. We want to show that

$$\mathsf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}).$$

Sampling and reparameterization: proof

1. Mean: By linearity of expectation,

$$\begin{aligned}\mathbb{E}[\mathsf{x}] &= \mathbb{E}[\boldsymbol{\mu} + \mathsf{L}\boldsymbol{\epsilon}] \\ &= \boldsymbol{\mu} + \mathsf{L}\mathbb{E}[\boldsymbol{\epsilon}] \\ &= \boxed{\boldsymbol{\mu}}.\end{aligned}$$

2. Covariance: By the definition of covariance,

$$\begin{aligned}\mathrm{Cov}(\mathsf{x}) &= \mathbb{E}\left[(\mathsf{x} - \boldsymbol{\mu})(\mathsf{x} - \boldsymbol{\mu})^T\right] \\ &= \mathbb{E}\left[\mathsf{L}\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T\mathsf{L}^T\right] \\ &= \mathsf{L}\mathbb{E}[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T]\mathsf{L}^T \\ &= \mathsf{L}\mathsf{I}\mathsf{L}^T \\ &= \boxed{\boldsymbol{\Sigma}}.\end{aligned}$$

Why is the resulting distribution Gaussian?

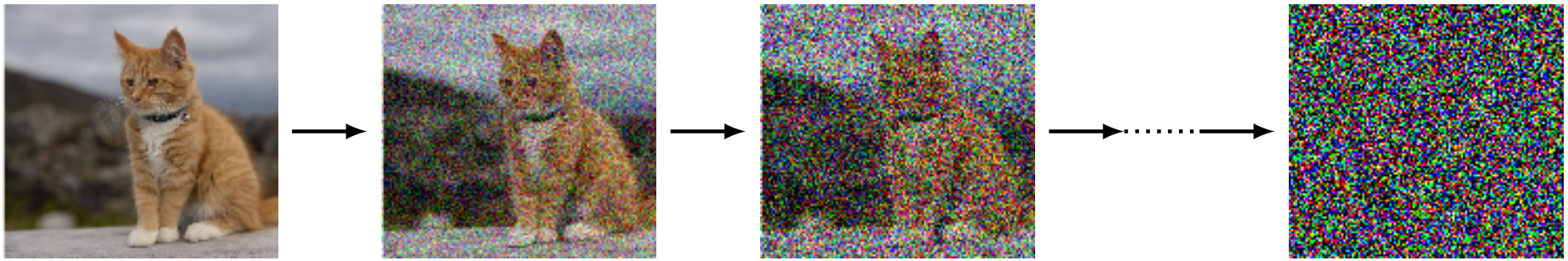
1. A fundamental property of Gaussian random vectors is that any affine transformation of a Gaussian random vector is also Gaussian.
2. Since ϵ is Gaussian and $x = \mu + L\epsilon$ is an affine transformation, x must also be Gaussian.
3. Combining this with its mean and covariance gives

$$\boxed{x \sim \mathcal{N}(\mu, \Sigma).}$$

4. For the isotropic case, simply take $L = \sigma I$:

$$\boxed{x = \mu + \sigma\epsilon, \qquad \epsilon \sim \mathcal{N}(0, I) \quad \Longrightarrow \quad x \sim \mathcal{N}(\mu, \sigma^2 I).}$$

Diffusion models: image to noise



The forward diffusion process begins with

$$\boxed{\mathsf{z}_1 = \sqrt{1-\beta_1}\mathsf{x} + \sqrt{\beta_1}\boldsymbol{\epsilon}_1}$$

1. $\mathsf{x} \in \mathbb{R}^D$ is an original training image
2. $\mathsf{z}_1 \in \mathbb{R}^D$ is the image after adding noise
3. $\boldsymbol{\epsilon}_1 \sim \mathcal{N}(0, \mathsf{I})$ is a vector of independent standard Gaussian noise
4. $\beta_1 \in (0, 1)$ controls how much noise is added

Why square roots?

$$\boxed{z_1 = \sqrt{1-\beta_1}x + \sqrt{\beta_1}\epsilon_1}$$

1. Suppose the original images have mean $\boldsymbol{\mu}_x$ and covariance $\boldsymbol{\Sigma}_x$.
2. Because the Gaussian noise is independent of the original image,

$$\mathbb{E}[z_1] = \sqrt{1-\beta_1}\boldsymbol{\mu}_x,$$

and

$$\text{Cov}(z_1) = (1-\beta_1)\boldsymbol{\Sigma}_x + \beta_1 I.$$

$$\text{Cov}(z_1) - I = (1-\beta_1)(\boldsymbol{\Sigma}_x - I).$$

3. The mean moves toward zero
4. The covariance moves toward the unit matrix

Transition distribution

$$\boxed{z_1 = \sqrt{1-\beta_1}x + \sqrt{\beta_1}\epsilon_1}$$

1. Consider

$$z_1 = a + B\epsilon_1,$$

where

$$a = \sqrt{1-\beta_1}x, \qquad B = \sqrt{\beta_1}I.$$

2. Since

$$\epsilon_1 \sim \mathcal{N}(0, I),$$

the affine transformation property of Gaussian distributions gives

$$z_1 \mid x \sim \mathcal{N}\left(a, BB^\top\right).$$

3. Substituting,

$$BB^\top = \beta_1 I$$

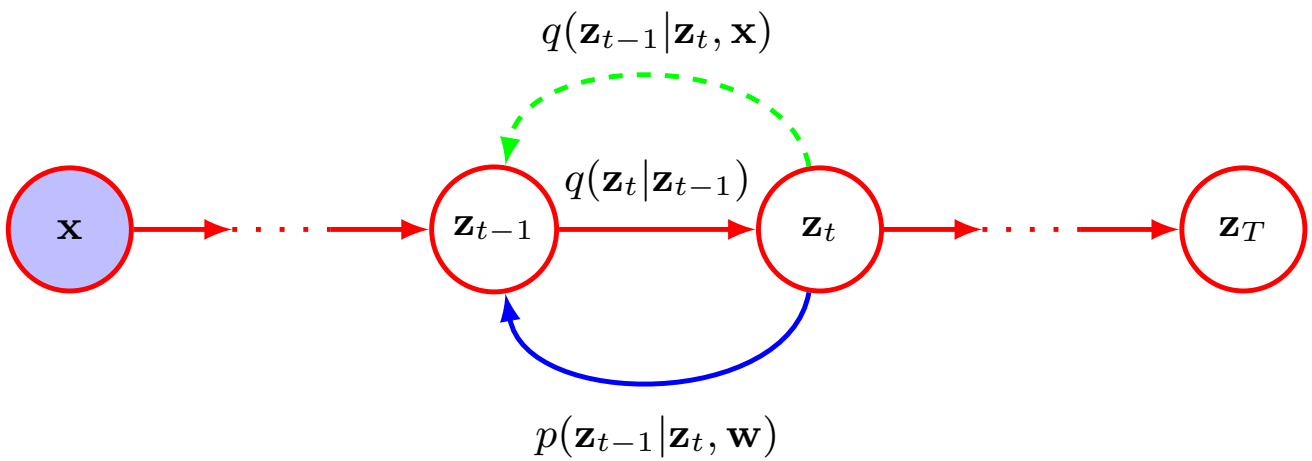
Gives

$$\boxed{q(z_1 \mid x) = \mathcal{N}\left(z_1 \mid \sqrt{1-\beta_1}x, \beta_1 I\right)}$$

Multiple steps

$$\mathsf{x} \to \mathsf{z}_1 \to \mathsf{z}_2 \to \cdots \to \mathsf{z}_T.$$

1. At step t,
$$\boxed{\mathsf{z}_t = \sqrt{1-\beta_t}\mathsf{z}_{t-1} + \sqrt{\beta_t}\boldsymbol{\epsilon}_t}$$
2. The corresponding transition distribution is
$$q(\mathsf{z}_t \mid \mathsf{z}_{t-1}) = \mathcal{N}\left(\sqrt{1-\beta_t}\mathsf{z}_{t-1}, \beta_t \mathsf{I}\right).$$
3. This forms a Markov chain:
$$q(\mathsf{z}_t \mid \mathsf{z}_{t-1}, \ldots, \mathsf{z}_1, \mathsf{x}) = q(\mathsf{z}_t \mid \mathsf{z}_{t-1}).$$



Joint distribution of latent variables

$$q(z_1, \ldots, z_t \mid x) = q(z_1 \mid x) \prod_{\tau=2}^{t} q(z_\tau \mid z_{\tau-1})$$

1. This follows directly from the chain rule of probability and the Markov property.
2. For example, when $t = 3$,

$$\begin{aligned} & q(z_1, z_2, z_3 \mid x) \\ =& q(z_1 \mid x) q(z_2 \mid z_1, x) q(z_3 \mid z_2, z_1, x) \\ =& q(z_1 \mid x) q(z_2 \mid z_1) q(z_3 \mid z_2). \end{aligned}$$

Diffusion kernel

$$\boxed{q(z_1, \ldots, z_t \mid x) = q(z_1 \mid x) \prod_{\tau=2}^{t} q(z_\tau \mid z_{\tau-1})}$$

1. The diffusion kernel is obtained by integrating out the intermediate variables:

$$q(z_t \mid x) = \int \cdots \int q(z_1, \ldots, z_t \mid x) \, dz_1 \cdots dz_{t-1}.$$

2. Because the transitions are linear Gaussian transformations, the resulting marginal distribution is also Gaussian.
3. We can derive its mean and covariance analytically.

$$\boxed{q(z_t \mid x) = \mathcal{N}\left(z_t \mid \sqrt{\alpha_t} x, (1 - \alpha_t) I\right)}$$

where

$$\alpha_t = \prod_{\tau=1}^{t} (1 - \beta_\tau).$$

Generating samples

$$\boxed{q(\mathsf{z}_t \mid \mathsf{x}) = \mathcal{N}\left(\mathsf{z}_t \mid \sqrt{\alpha_t}\mathsf{x}, (1-\alpha_t)\mathsf{I}\right)}$$

1. For a Gaussian random vector

$$\mathsf{z} \sim \mathcal{N}(\boldsymbol{\mu}, \sigma^2\mathsf{I}),$$

we can generate a sample using

$$\mathsf{z} = \boldsymbol{\mu} + \sigma\boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathsf{I}).$$

2. Substituting

$$\boldsymbol{\mu} = \sqrt{\alpha_t}\mathsf{x}, \qquad \sigma = \sqrt{1-\alpha_t},$$

Gives

$$\boxed{\mathsf{z}_t = \sqrt{\alpha_t}\mathsf{x} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_t}$$

where

$$\boldsymbol{\epsilon}_t \sim \mathcal{N}(0, \mathsf{I}).$$

Infinite steps

$$\boxed{z_t = \sqrt{1-\beta_t} z_{t-1} + \sqrt{\beta_t}\epsilon_t}$$

Here, ϵ_t represents the **new noise introduced at step** t.

$$\boxed{z_t = \sqrt{\alpha_t} x + \sqrt{1-\alpha_t}\epsilon_t}$$

Here, ϵ_t represents the **aggregate effect of all Gaussian noise** introduced from steps 1 through t.

If the noise schedule is such that

$$\alpha_T \to 0 \qquad \text{as } T \to \infty,$$

then the direct sampling equation becomes

$$z_T \sqrt{\alpha_T} x + \sqrt{1-\alpha_T}\epsilon_T \longrightarrow \epsilon_T.$$

$$\boxed{q(z_T \mid x) = \mathcal{N}(z_T \mid 0, I)} \quad T \to +\infty$$

$$\alpha_T = \prod_{t=1}^{T} (1-\beta_t).$$

Noise schedule

1. **An important mathematical qualification:** The condition $0 < \beta_t < 1$ alone does not guarantee that $\alpha_T \to 0$. We need

$$\prod_{t=1}^{\infty}(1-\beta_t) = 0.$$

2. For example, a sufficient condition is

$$\sum_{t=1}^{\infty}\beta_t = \infty.$$

This condition ensures that enough cumulative noise is introduced to eliminate the original signal.

Reverse distribution

Given z_t, we want to infer z_{t-1}

$$\boxed{q(z_{t-1} \mid z_t) = \frac{q(z_t \mid z_{t-1})q(z_{t-1})}{q(z_t)}}$$

$$\boxed{q(z_{t-1}) = \int q(z_{t-1} \mid x)p(x)\,dx}$$

1. $p(x)$ is the unknown distribution of real images
2. $q(z_{t-1} \mid x)$ is the Gaussian diffusion kernel derived previously
3. The integral is intractable

Reverse conditional distribution

$$\boxed{q(z_{t-1} \mid z_t, x)}$$

1. Given both the original image x and its noisy version z_t, what is the distribution of the intermediate image z_{t-1}?
2. The additional information x makes the problem tractable.

$$\boxed{q(z_{t-1} \mid z_t, x) = \frac{q(z_t \mid z_{t-1}, x) q(z_{t-1} \mid x)}{q(z_t \mid x)}}$$

3. Because the forward process is a Markov chain,

$$\boxed{q(z_t \mid z_{t-1}, x) = q(z_t \mid z_{t-1})}$$

4. Therefore,

$$q(z_{t-1} \mid z_t, x) \propto q(z_t \mid z_{t-1}) q(z_{t-1} \mid x),$$

Both factors on the right-hand side are Gaussian. Their product is proportional to another Gaussian, which means we can derive the reverse conditional distribution analytically.

Reverse conditional distribution

Using the technique of completing the square, we can obtain

$$\boxed{q(\mathsf{z}_{t-1} \mid \mathsf{z}_t, \mathsf{x}) = \mathcal{N}\left(\mathsf{z}_{t-1} \mid \mathsf{m}_t(\mathsf{x}, \mathsf{z}_t), \sigma_t^2 \mathsf{I}\right)}$$

where

$$\boxed{\mathsf{m}_t(\mathsf{x}, \mathsf{z}_t) = \frac{(1-\alpha_{t-1})\sqrt{1-\beta_t}\mathsf{z}_t + \sqrt{\alpha_{t-1}}\beta_t \mathsf{x}}{1-\alpha_t}}$$

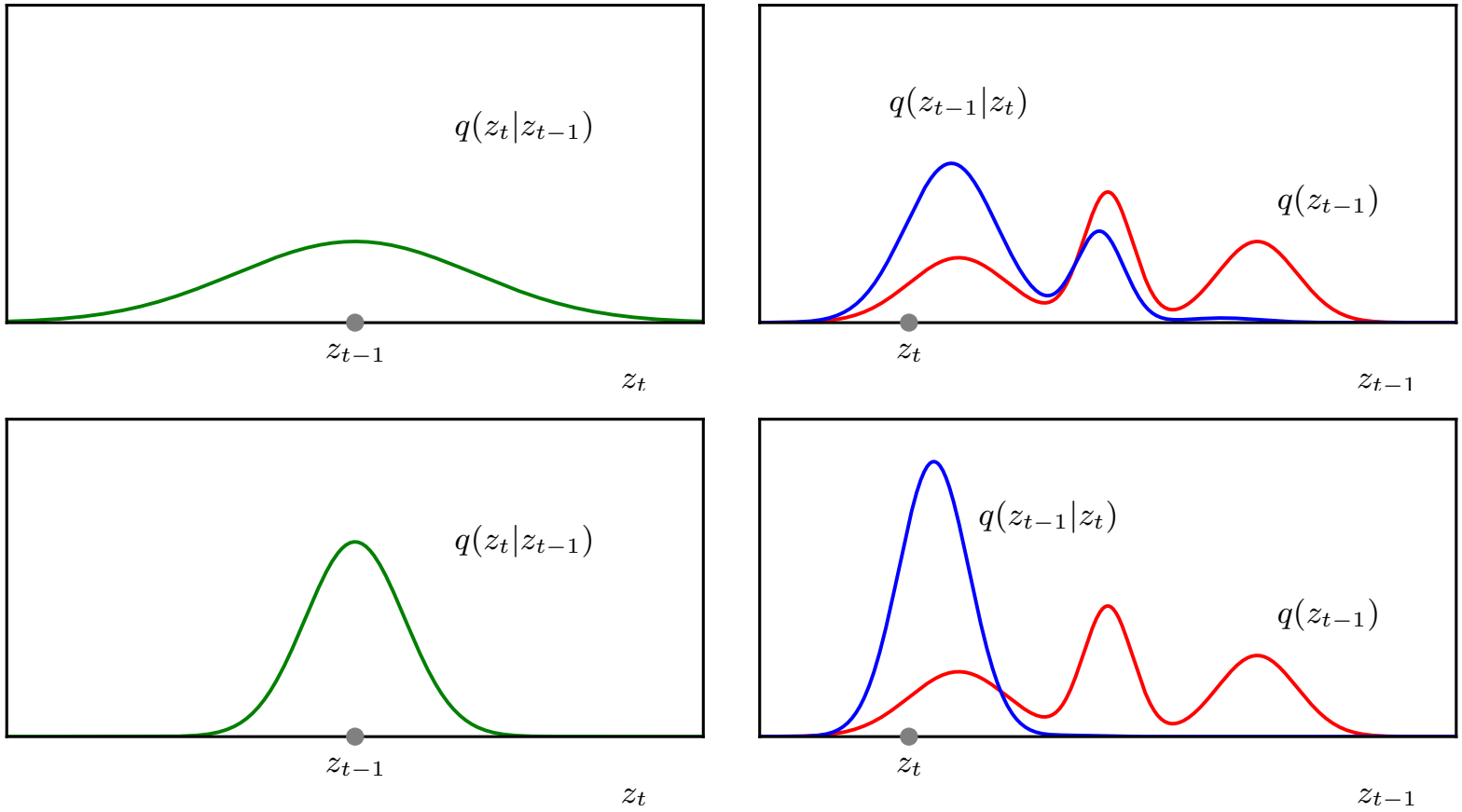
and

$$\boxed{\sigma_t^2 = \frac{\beta_t(1-\alpha_{t-1})}{1-\alpha_t}}$$

Learning the reverse distribution

1. The forward encoder model is defined by a sequence of Gaussian conditional distributions $q(\mathrm{z}_t \mid \mathrm{z}_{t-1})$
2. Inverting this directly leads to a distribution $q(\mathrm{z}_{t-1} \mid \mathrm{z}_t)$ that is intractable
3. Instead, we learn an approximation to the reverse distribution by using a distribution $p(\mathrm{z}_{t-1} \mid \mathrm{z}_t, \mathrm{w})$ governed by a deep neural network
4. Once the network is trained, we can sample from the simple Gaussian distribution over z_T and transform it into a sample from the data distribution $p(\mathrm{x})$ through a sequence of reverse sampling steps by repeated application of the trained network

More on the reverse distribution



$$q(z_{t-1} \mid z_t) = \frac{q(z_t \mid z_{t-1})q(z_{t-1})}{q(z_t)}$$

1. If $\beta_t \ll 1$ then $q(z_{t-1} \mid z_t)$ will be approximately a Gaussian over z_{t-1}
2. Since the variances are small, T needs to be large (several thousand)

Learning the reverse distribution

1. For small variance, the reverse distribution $q(\mathsf{z}_{t-1} \mid \mathsf{z}_t)$ will have a covariance that is close to the covariance $\beta_t \mathsf{I}$ of the forward noise process $q(\mathsf{z}_t \mid \mathsf{z}_{t-1})$, so we have

$$\boxed{p(\mathsf{z}_{t-1} \mid \mathsf{z}_t, \mathsf{w}) = \mathcal{N}\left(\mathsf{z}_{t-1} \mid \boldsymbol{\mu}(\mathsf{z}_t, \mathsf{w}, t), \beta_t \mathsf{I}\right)}$$

   where $\boldsymbol{\mu}(\mathsf{z}_t, \mathsf{w}, t)$ is a deep neural network with parameters w
2. The network receives two inputs:

$$(\mathsf{z}_t, t)$$

   and predicts

$$\boldsymbol{\mu}(\mathsf{z}_t, \mathsf{w}, t).$$

3. The covariance is fixed to $\beta_t \mathsf{I}$, while the mean is learned
4. The noise variance β_t generally varies across diffusion steps
5. The network must therefore know the current diffusion step to determine how much denoising is required
6. By providing t explicitly, the same network can be used for all reverse steps

Generating new samples

$$p(\mathrm{x}, \mathrm{z}_1, \ldots, \mathrm{z}_T \mid \mathrm{w}) = p(\mathrm{z}_T) \left\{ \prod_{t=2}^{T} p(\mathrm{z}_{t-1} \mid \mathrm{z}_t, \mathrm{w}) \right\} p(\mathrm{x} \mid \mathrm{z}_1, \mathrm{w})$$

1. For the four variables $\mathrm{x}, \mathrm{z}_1, \mathrm{z}_2, \mathrm{z}_3$, we can always write

$$\begin{aligned} & p(\mathrm{x}, \mathrm{z}_1, \mathrm{z}_2, \mathrm{z}_3 \mid \mathrm{w}) \\ &= p(\mathrm{z}_3 \mid \mathrm{w}) \\ &\quad \times p(\mathrm{z}_2 \mid \mathrm{z}_3, \mathrm{w}) \\ &\quad \times p(\mathrm{z}_1 \mid \mathrm{z}_2, \mathrm{z}_3, \mathrm{w}) \\ &\quad \times p(\mathrm{x} \mid \mathrm{z}_1, \mathrm{z}_2, \mathrm{z}_3, \mathrm{w}). \end{aligned}$$

2. Apply the Markov assumption,

$$\begin{aligned} & p(\mathrm{x}, \mathrm{z}_1, \mathrm{z}_2, \mathrm{z}_3 \mid \mathrm{w}) \\ &= p(\mathrm{z}_3) p(\mathrm{z}_2 \mid \mathrm{z}_3, \mathrm{w}) \\ &\quad \times p(\mathrm{z}_1 \mid \mathrm{z}_2, \mathrm{w}) p(\mathrm{x} \mid \mathrm{z}_1, \mathrm{w}). \end{aligned}$$

The initial noise distribution $p(\mathrm{z}_3)$ does not depend on w.

Generating new samples

$$\boxed{p(\mathsf{x}, \mathsf{z}_1, \ldots, \mathsf{z}_T \mid \mathsf{w}) = p(\mathsf{z}_T) \left\{ \prod_{t=2}^{T} p(\mathsf{z}_{t-1} \mid \mathsf{z}_t, \mathsf{w}) \right\} p(\mathsf{x} \mid \mathsf{z}_1, \mathsf{w})}$$

1. $\mathsf{z}_T \sim \mathcal{N}(0, \mathsf{I})$.
2. For $t = T, T-1, \ldots, 2$, sample

$$\mathsf{z}_{t-1} \sim \mathcal{N}\left(\boldsymbol{\mu}(\mathsf{z}_t, \mathsf{w}, t), \beta_t \mathsf{I}\right).$$

Equivalently,

$$\boxed{\mathsf{z}_{t-1} = \boldsymbol{\mu}(\mathsf{z}_t, \mathsf{w}, t) + \sqrt{\beta_t}\boldsymbol{\epsilon}_t} \quad \boldsymbol{\epsilon}_t \sim \mathcal{N}(0, \mathsf{I}).$$

3.

$$\mathsf{x} \sim p(\mathsf{x} \mid \mathsf{z}_1, \mathsf{w}).$$

The resulting x is a sample from the learned generative distribution

Training the reverse decoder

$$p(\mathrm{x}, \mathrm{z}_1, \ldots, \mathrm{z}_T \mid \mathrm{w}) = p(\mathrm{z}_T) \left[\prod_{t=2}^{T} p(\mathrm{z}_{t-1} \mid \mathrm{z}_t, \mathrm{w}) \right] p(\mathrm{x} \mid \mathrm{z}_1, \mathrm{w}).$$

1. For a given data point x, the likelihood is obtained by marginalizing out all latent variables:

$$\boxed{p(\mathrm{x} \mid \mathrm{w}) = \int \cdots \int p(\mathrm{x}, \mathrm{z}_1, \ldots, \mathrm{z}_T \mid \mathrm{w})\, d\mathrm{z}_1 \cdots d\mathrm{z}_T.}$$

2. Ideally, we would maximize the log likelihood over the training dataset:

$$\max_{\mathrm{w}} \sum_{n=1}^{N} \ln p(\mathrm{x}_n \mid \mathrm{w})$$

3. The likelihood involves integrating over all possible trajectories by which noise samples could give rise to the observed data point
4. Intractable as they involve integrating over the highly complex neural network functions

Variational inference

1. For training, we need $p(\mathrm{x} \mid \mathrm{w})$, which is intractable.
2. Instead, we have $p(\mathrm{x}, \mathrm{z} \mid \mathrm{w})$, where $\mathrm{z} = (\mathrm{z}_1, \ldots, \mathrm{z}_T)$
3. Introduce a variational distribution $q(\mathrm{z})$
4. For any valid distribution $q(\mathrm{z})$

$$\ln p(\mathrm{x} \mid \mathrm{w}) = \underbrace{\int q(\mathrm{z}) \ln \frac{p(\mathrm{x}, \mathrm{z} \mid \mathrm{w})}{q(\mathrm{z})} \, d\mathrm{z}}_{\mathcal{L}(\mathrm{w})} + \mathrm{KL}\left(q(\mathrm{z}) \,\|\, p(\mathrm{z} \mid \mathrm{x}, \mathrm{w})\right)$$

5. $\mathcal{L}(\mathrm{w})$ is called the *evidence lower bound* (ELBO)
6. Since $\mathrm{KL}(\cdot\|\cdot) \geq 0$, it follows that

$$\ln p(\mathrm{x} \mid \mathrm{w}) \geq \mathcal{L}(\mathrm{w})$$

7. We train the neural network by maximizing the lower bound $\mathcal{L}(\mathrm{w})$

Review: Kullback–Leibler divergence

1. Kullback–Leibler divergence provides a way of quantifying the difference between two distributions.
2. KL divergence is defined as

$$\mathrm{KL}(q(\mathsf{z})\|p(\mathsf{z})) = \mathbb{E}_{q(\mathsf{z})}\left[\log \frac{q(\mathsf{z})}{p(\mathsf{z})}\right] = -\mathbb{E}_{q(\mathsf{z})}\left[\log \frac{p(\mathsf{z})}{q(\mathsf{z})}\right]$$

3. KL divergence is
   - Non-negative: $\mathrm{KL}(q(\mathsf{z})\|p(\mathsf{z})) \geq 0$ for any $q(\mathsf{z})$ and $p(\mathsf{z})$;
   - $\mathrm{KL}(q(\mathsf{z})\|p(\mathsf{z})) = 0$ if and only if $q(\mathsf{z}) = p(\mathsf{z})$ almost everywhere;
   - Not symmetric: in general, $\mathrm{KL}(q(\mathsf{z})\|p(\mathsf{z})) \neq \mathrm{KL}(p(\mathsf{z})\|q(\mathsf{z}))$.

The evidence lower bound (ELBO)

1. We are free to choose any form we like for $q(\mathsf{z})$ as long as it is a valid probability distribution, i.e., that it is non-negative and integrates to 1
2. Diffusion models choose $q(\mathsf{z})$ to be the *fixed* distribution $q(\mathsf{z}_1, \ldots, \mathsf{z}_T \mid \mathsf{x})$ that depends on x
3. Substitute $q(\mathsf{z}) = q(\mathsf{z}_1, \ldots, \mathsf{z}_T \mid \mathsf{x})$ and $p(\mathsf{x}, \mathsf{z}_1, \ldots, \mathsf{z}_T \mid \mathsf{w})$ into

$$\mathcal{L}(\mathsf{w}) = \int q(\mathsf{z}) \ln \frac{p(\mathsf{x}, \mathsf{z} \mid \mathsf{w})}{q(\mathsf{z})} \, d\mathsf{z}$$

The evidence lower bound (ELBO)

$$q(z_1, \ldots, z_t \mid x) = q(z_1 \mid x) \prod_{\tau=2}^{t} q(z_\tau \mid z_{\tau-1})$$

$$p(x, z_1, \ldots, z_T \mid w) = p(z_T) \left\{ \prod_{t=2}^{T} p(z_{t-1} \mid z_t, w) \right\} p(x \mid z_1, w)$$

$$\mathcal{L}(w) = \int q(z) \ln \frac{p(x, z \mid w)}{q(z)} \, dz = \mathbb{E}_q \left[\ln \frac{p(x, z \mid w)}{q(z)} \right]$$

$$= \mathbb{E}_q \left[\ln \frac{p(z_T) \left\{ \prod_{t=2}^{T} p(z_{t-1} \mid z_t, w) \right\} p(x \mid z_1, w)}{q(z_1 \mid x) \prod_{t=2}^{T} q(z_t \mid z_{t-1}, x)} \right]$$

$$= \mathbb{E}_q \left[\ln p(z_T) + \sum_{t=2}^{T} \ln \frac{p(z_{t-1} \mid z_t, w)}{q(z_t \mid z_{t-1}, x)} - \ln q(z_1 \mid x) + \ln p(x \mid z_1, w) \right]$$

where $\mathbb{E}_q[\cdot] \equiv \int \cdots \int q(z_1 \mid x) \prod_{t=2}^{T} q(z_t \mid z_{t-1})[\cdot] \, dz_1 \cdots dz_T$.

The reconstruction term

$$\mathcal{L}(\mathrm{w}) = \mathbb{E}_q \left[\ln p(\mathrm{z}_T) + \sum_{t=2}^{T} \ln \frac{p(\mathrm{z}_{t-1} \mid \mathrm{z}_t, \mathrm{w})}{q(\mathrm{z}_t \mid \mathrm{z}_{t-1}, \mathrm{x})} - \ln q(\mathrm{z}_1 \mid \mathrm{x}) + \ln p(\mathrm{x} \mid \mathrm{z}_1, \mathrm{w}) \right]$$

1. The first and third terms do not depend on w
2. The fourth term is the **reconstruction term** and measures how well the reverse model can reconstruct x from z_1

$$\mathcal{L}_{\mathrm{rec}}(\mathrm{w}) = \mathbb{E}_{q(\mathrm{z}_1|\mathrm{x})} [\ln p(\mathrm{x} \mid \mathrm{z}_1, \mathrm{w})] = \int q(\mathrm{z}_1 \mid \mathrm{x}) \ln p(\mathrm{x} \mid \mathrm{z}_1, \mathrm{w}) \, d\mathrm{z}_1$$

3. $p(\mathrm{x} \mid \mathrm{z}_1, \mathrm{w})$ is the learned reverse distribution
4. Draw L independent samples:

$$\mathrm{z}_1^{(1)}, \ldots, \mathrm{z}_1^{(L)} \sim q(\mathrm{z}_1 \mid \mathrm{x}).$$

Then approximate the expectation using

$$\mathcal{L}_{\mathrm{rec}}(\mathrm{w}) \approx \frac{1}{L} \sum_{l=1}^{L} \ln p(\mathrm{x} \mid \mathrm{z}_1^{(l)}, \mathrm{w})$$

Sampling two latent variables

$$\mathcal{L}(\mathsf{w}) = \mathbb{E}_q \left[\ln p(\mathsf{z}_T) + \sum_{t=2}^{T} \ln \frac{p(\mathsf{z}_{t-1} \mid \mathsf{z}_t, \mathsf{w})}{q(\mathsf{z}_t \mid \mathsf{z}_{t-1}, \mathsf{x})} - \ln q(\mathsf{z}_1 \mid \mathsf{x}) + \ln p(\mathsf{x} \mid \mathsf{z}_1, \mathsf{w}) \right]$$

1. The second term involves

$$\mathbb{E}_q \left[\ln \frac{p(\mathsf{z}_{t-1} \mid \mathsf{z}_t, \mathsf{w})}{q(\mathsf{z}_t \mid \mathsf{z}_{t-1}, \mathsf{x})} \right].$$

2. This expectation involves both z_{t-1} and z_t.
3. One could sample z_{t-1} from $q(\mathsf{z}_{t-1} \mid \mathsf{x})$, then sample z_t from $q(\mathsf{z}_t \mid \mathsf{z}_{t-1})$.
4. However, the resulting Monte Carlo estimate can have high variance.
5. The goal is to **analytically integrate out** z_{t-1} instead of sampling it.

Conditional reverse distribution

Using Bayes' theorem,

$$q(z_t \mid z_{t-1}, x) = \frac{q(z_{t-1} \mid z_t, x) q(z_t \mid x)}{q(z_{t-1} \mid x)}$$

1. $q(z_{t-1} \mid z_t, x)$: the exact reverse conditional distribution when the original observation x is known.
2. Although the unconditional reverse distribution $q(z_{t-1} \mid z_t)$ is generally intractable, the **conditional reverse distribution $q(z_{t-1} \mid z_t, x)$ is Gaussian and can be calculated analytically**.
3. This is possible because conditioning on x makes the forward process a linear-Gaussian model.

New ELBO

$$\boxed{q(z_t \mid z_{t-1}, x) = \frac{q(z_{t-1} \mid z_t, x) q(z_t \mid x)}{q(z_{t-1} \mid x)}}$$

$$\ln \frac{p(z_{t-1} \mid z_t, w)}{q(z_t \mid z_{t-1}, x)} = \ln \frac{p(z_{t-1} \mid z_t, w) q(z_{t-1} \mid x)}{q(z_{t-1} \mid z_t, x) q(z_t \mid x)}$$

$$= \boxed{\ln \frac{p(z_{t-1} \mid z_t, w)}{q(z_{t-1} \mid z_t, x)} + \ln \frac{q(z_{t-1} \mid x)}{q(z_t \mid x)}}.$$

$$\boxed{\mathcal{L}(w) \stackrel{c}{=} \mathbb{E}_q \left[\sum_{t=2}^{T} \ln \frac{p(z_{t-1} \mid z_t, w)}{q(z_{t-1} \mid z_t, x)} + \ln p(x \mid z_1, w) \right]}$$

This objective has two components:

1. A reconstruction term for generating x from z_1.
2. A collection of terms comparing the learned reverse transitions with the exact conditional reverse transitions.

More on the second term

$$\mathbb{E}_q \left[\ln \frac{p(z_{t-1} \mid z_t, w)}{q(z_{t-1} \mid z_t, x)} \right].$$

1. Using the law of iterated expectations,

$$\mathbb{E}_q \left[\ln \frac{p(z_{t-1} \mid z_t, w)}{q(z_{t-1} \mid z_t, x)} \right] = \mathbb{E}_{q(z_t|x)} \left[\mathbb{E}_{q(z_{t-1}|z_t,x)} \left[\ln \frac{p(z_{t-1} \mid z_t, w)}{q(z_{t-1} \mid z_t, x)} \right] \right].$$

2. Recall the definition

$$\mathrm{KL}(q\|p) = \mathbb{E}_q \left[\ln \frac{q}{p} \right] = -\mathbb{E}_q \left[\ln \frac{p}{q} \right].$$

3. Therefore,

$$\mathbb{E}_{q(z_{t-1}|z_t,x)} \left[\ln \frac{p(z_{t-1} \mid z_t, w)}{q(z_{t-1} \mid z_t, x)} \right] = -\mathrm{KL}\left(q(z_{t-1} \mid z_t, x) \| p(z_{t-1} \mid z_t, w) \right).$$

The final objective

$$\mathcal{L}(\mathrm{w}) \stackrel{c}{=} -\sum_{t=2}^{T} \mathbb{E}_{q(\mathrm{z}_t|\mathrm{x})} \left[\mathrm{KL}\left(q(\mathrm{z}_{t-1} \mid \mathrm{z}_t, \mathrm{x}) \| p(\mathrm{z}_{t-1} \mid \mathrm{z}_t, \mathrm{w})\right)\right] + \mathbb{E}_q \left[\ln p(\mathrm{x} \mid \mathrm{z}_1, \mathrm{w})\right]$$

$$\mathcal{L}(\mathrm{w}) \stackrel{c}{=} \underbrace{\int q(\mathrm{z}_1 \mid \mathrm{x}) \ln p(\mathrm{x} \mid \mathrm{z}_1, \mathrm{w}) \, d\mathrm{z}_1}_{\text{Reconstruction term}}$$

$$\underbrace{-\sum_{t=2}^{T} \int q(\mathrm{z}_t \mid \mathrm{x}) \mathrm{KL}\left(q(\mathrm{z}_{t-1} \mid \mathrm{z}_t, \mathrm{x}) \| p(\mathrm{z}_{t-1} \mid \mathrm{z}_t, \mathrm{w})\right) \, d\mathrm{z}_t}_{\text{Consistency terms}} .$$

1. The reconstruction term encourages the network to recover the original data from a slightly noisy version
2. The consistency terms encourage the learned reverse transitions to agree with the exact conditional reverse transitions

Consistency terms

$$q(\mathsf{z}_{t-1} \mid \mathsf{z}_t, \mathsf{x}) = \mathcal{N}\left(\mathsf{z}_{t-1} \mid \mathsf{m}_t(\mathsf{x}, \mathsf{z}_t), \tilde{\beta}_t \mathsf{I}\right).$$
$$p(\mathsf{z}_{t-1} \mid \mathsf{z}_t, \mathsf{w}) = \mathcal{N}\left(\mathsf{z}_{t-1} \mid \boldsymbol{\mu}(\mathsf{z}_t, \mathsf{w}, t), \beta_t \mathsf{I}\right).$$

1. The two Gaussian distributions have different means
2. Their covariance matrices are fixed and do not depend on w

$$\boxed{\begin{aligned}&\mathrm{KL}\left(q(\mathsf{z}_{t-1} \mid \mathsf{z}_t, \mathsf{x}) \| p(\mathsf{z}_{t-1} \mid \mathsf{z}_t, \mathsf{w})\right) \\ &\qquad = \frac{1}{2\beta_t} \left\|\mathsf{m}_t(\mathsf{x}, \mathsf{z}_t) - \boldsymbol{\mu}(\mathsf{z}_t, \mathsf{w}, t)\right\|^2 + \text{const.}\end{aligned}}$$

3. The KL divergence reduces to a weighted squared difference between the true conditional posterior mean and the neural network's predicted mean
4. The exact posterior covariance $\tilde{\beta}_t \mathsf{I}$ need not equal the model covariance $\beta_t \mathsf{I}$. Their difference contributes only to the constant because both are fixed.

Sampling of one variable

$$\mathcal{L}_t(\mathsf{w}) = -\,\mathbb{E}_{q(\mathsf{z}_t|\mathsf{x})}\left[\mathrm{KL}\left(q(\mathsf{z}_{t-1} \mid \mathsf{z}_t, \mathsf{x}) \| p(\mathsf{z}_{t-1} \mid \mathsf{z}_t, \mathsf{w})\right)\right]$$
$$\stackrel{c}{=} -\,\frac{1}{2\beta_t}\mathbb{E}_{q(\mathsf{z}_t|\mathsf{x})}\left[\|\mathsf{m}_t(\mathsf{x}, \mathsf{z}_t) - \boldsymbol{\mu}(\mathsf{z}_t, \mathsf{w}, t)\|^2\right]$$

1. The only remaining expectation is over z_t.

$$q(\mathsf{z}_t \mid \mathsf{x}) = \mathcal{N}\left(\mathsf{z}_t \mid \sqrt{\bar{\alpha}_t}\mathsf{x}, (1-\bar{\alpha}_t)\mathsf{I}\right),$$

2. Sample directly:

$$\boxed{\mathsf{z}_t = \sqrt{\bar{\alpha}_t}\mathsf{x} + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathsf{I}).}$$

3. Calculate the loss

$$\boxed{\ell_t(\mathsf{w}) = \frac{1}{2\beta_t}\|\mathsf{m}_t(\mathsf{x}, \mathsf{z}_t) - \boldsymbol{\mu}(\mathsf{z}_t, \mathsf{w}, t)\|^2.}$$

4. Monte Carlo estimate using only one sampled noisy state z_t.
5. No need to sample z_{t-1} as its expectation has already been evaluated analytically through the Gaussian KL divergence.

The final procedure

1. Given x and a sampled z_t, the target mean can be computed exactly.

$$q(z_{t-1} \mid z_t, x) = \mathcal{N}\left(m_t(x, z_t), \tilde{\beta}_t I\right)$$

2. The neural network predicts

$$\mu(z_t, w, t).$$

3. The training objective is to make this prediction match the analytical target.
4. This is essentially a supervised regression problem:

Input to neural network	**Target**	**Loss**
z_t, t	$m_t(x, z_t)$	Weighted MSE

5. The network does not receive the clean x as an input.
6. The clean observation is used to construct the training target, while the network must learn to predict that target from the noisy observation.

Predicting the noise

$$z_t = \sqrt{\alpha_t} x + \sqrt{1-\alpha_t}\epsilon_t,$$

We can solve for x:

$$\boxed{x = \frac{1}{\sqrt{\alpha_t}} z_t - \frac{\sqrt{1-\alpha_t}}{\sqrt{\alpha_t}} \epsilon_t.}$$

1. If we know z_t and ϵ_t, we can recover the original observation exactly.
2. During generation, the noise is unknown. The neural network will learn to estimate it.
3. If the network predicts

   $$\widehat{\epsilon}_t = g(z_t, w, t),$$

   we can estimate the original observation by

   $$\boxed{\widehat{x} = \frac{z_t - \sqrt{1-\alpha_t} g(z_t, w, t)}{\sqrt{\alpha_t}}.}$$

4. Predicting the noise is mathematically equivalent to predicting the clean observation for a fixed z_t and t.

Conditional reverse mean and the noise

1. Recall the conditional reverse distribution

$$q(z_{t-1} \mid z_t, x) = \mathcal{N}\left(m_t(x, z_t), \tilde{\beta}_t I\right),$$

$$m_t(x, z_t) = \frac{\sqrt{\alpha_{t-1}}\beta_t}{1-\alpha_t} x + \frac{\sqrt{1-\beta_t}(1-\alpha_{t-1})}{1-\alpha_t} z_t.$$

2. Substituting x and simplifying gives

$$\boxed{m_t(x, z_t) = \frac{1}{\sqrt{1-\beta_t}}\left(z_t - \frac{\beta_t}{\sqrt{1-\alpha_t}}\epsilon_t\right).}$$

3. The exact reverse mean can be expressed entirely in terms of z_t, the noise ϵ_t and the known diffusion coefficients α_t and β_t.
4. Instead of predicting the reverse mean directly, the network can predict the noise and then use this formula to construct the reverse mean.

Predicting the conditional reverse mean versus the noise

1. So far, the neural network directly predicts the mean $\boldsymbol{\mu}(\mathsf{z}_t, \mathsf{w}, t)$.

$$p(\mathsf{z}_{t-1} \mid \mathsf{z}_t, \mathsf{w}) = \mathcal{N}\left(\boldsymbol{\mu}(\mathsf{z}_t, \mathsf{w}, t), \beta_t \mathsf{I}\right).$$

2. Now introduce a different neural network: $\mathsf{g}(\mathsf{z}_t, \mathsf{w}, t)$, which predicts the <u>total</u> effective noise $\boldsymbol{\epsilon}_t$.
3. Then compute the predicted reverse mean by

$$\boxed{\boldsymbol{\mu}(\mathsf{z}_t, \mathsf{w}, t) = \frac{1}{\sqrt{1-\beta_t}}\left(\mathsf{z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}}\mathsf{g}(\mathsf{z}_t, \mathsf{w}, t)\right).}$$

4. The network does not directly output the reverse mean anymore. Instead, the computation is

$$\mathsf{z}_t \xrightarrow{\mathsf{g}(\cdot,\mathsf{w},t)} \widehat{\boldsymbol{\epsilon}}_t \xrightarrow{\text{Above equation}} \boldsymbol{\mu}(\mathsf{z}_t, \mathsf{w}, t).$$

5. This is a reparameterization of the reverse mean in terms of predicted noise.
6. It changes what the network is asked to predict and how its training loss is represented.

KL divergence objectives

1. Substituting $\mathrm{m}_t(\mathrm{x}, \mathrm{z}_t)$ and $\boldsymbol{\mu}(\mathrm{z}_t, \mathrm{w}, t)$ from previous two pages

$$\|\mathrm{m}_t(\mathrm{x}, \mathrm{z}_t) - \boldsymbol{\mu}(\mathrm{z}_t, \mathrm{w}, t)\|^2 = \frac{\beta_t^2}{(1-\alpha_t)(1-\beta_t)} \|\mathrm{g}(\mathrm{z}_t, \mathrm{w}, t) - \boldsymbol{\epsilon}_t\|^2 .$$

2. Substituting into the Gaussian KL divergence gives

$$\boxed{\begin{aligned} &\mathrm{KL}\left(q(\mathrm{z}_{t-1} \mid \mathrm{z}_t, \mathrm{x}) \| p(\mathrm{z}_{t-1} \mid \mathrm{z}_t, \mathrm{w})\right) \\ &= \frac{\beta_t}{2(1-\alpha_t)(1-\beta_t)} \|\mathrm{g}(\mathrm{z}_t, \mathrm{w}, t) - \boldsymbol{\epsilon}_t\|^2 + \text{const.} \end{aligned}}$$

3. Substituting $\mathrm{z}_t = \sqrt{\alpha_t}\mathrm{x} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_t$,

$$\boxed{\begin{aligned} \mathrm{KL} = &\frac{\beta_t}{2(1-\alpha_t)(1-\beta_t)} \\ &\times \left\|\mathrm{g}\left(\sqrt{\alpha_t}\mathrm{x} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_t, \mathrm{w}, t\right) - \boldsymbol{\epsilon}_t\right\|^2 + \text{const.} \end{aligned}}$$

4. Minimizing the KL divergence between the exact conditional reverse distribution and the learned reverse distribution is equivalent to minimizing a weighted squared error between the predicted noise and the actual noise.

The reconstruction term

1. The ELBO contains a reconstruction term at $t = 1$:

$$\mathbb{E}_{q(\mathsf{z}_1|\mathsf{x})}\left[\ln p(\mathsf{x} \mid \mathsf{z}_1, \mathsf{w})\right].$$

$$\text{where} \quad p(\mathsf{x} \mid \mathsf{z}_1, \mathsf{w}) = \mathcal{N}\left(\mathsf{x} \mid \boldsymbol{\mu}(\mathsf{z}_1, \mathsf{w}, 1), \beta_1 \mathsf{I}\right).$$

$$\boxed{\ln p(\mathsf{x} \mid \mathsf{z}_1, \mathsf{w}) = -\frac{1}{2\beta_1}\left\|\mathsf{x} - \boldsymbol{\mu}(\mathsf{z}_1, \mathsf{w}, 1)\right\|^2 + \text{const.}}$$

2. Now use

$$\mathsf{z}_1 = \sqrt{1-\beta_1}\mathsf{x} + \sqrt{\beta_1}\boldsymbol{\epsilon}_1$$

and

$$\boldsymbol{\mu}(\mathsf{z}_1, \mathsf{w}, 1) = \frac{1}{\sqrt{1-\beta_1}}\left(\mathsf{z}_1 - \sqrt{\beta_1}\mathsf{g}(\mathsf{z}_1, \mathsf{w}, 1)\right).$$

Subtracting,

$$\mathsf{x} - \boldsymbol{\mu}(\mathsf{z}_1, \mathsf{w}, 1) = \frac{\sqrt{\beta_1}}{\sqrt{1-\beta_1}}\left[\mathsf{g}(\mathsf{z}_1, \mathsf{w}, 1) - \boldsymbol{\epsilon}_1\right].$$

The reconstruction term

Therefore,

$$\boxed{\ln p(\mathsf{x} \mid \mathsf{z}_1, \mathsf{w}) = -\frac{1}{2(1-\beta_1)} \left\|\mathsf{g}(\mathsf{z}_1, \mathsf{w}, 1) - \boldsymbol{\epsilon}_1\right\|^2 + \text{const.}}$$

1. For $t = 1$, $\alpha_1 = 1 - \beta_1$. The weight in

$$\boxed{\begin{aligned}\mathrm{KL} = &\frac{\beta_t}{2(1-\alpha_t)(1-\beta_t)} \\ &\times \left\|\mathsf{g}\left(\sqrt{\alpha_t}\mathsf{x} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_t, \mathsf{w}, t\right) - \boldsymbol{\epsilon}_t\right\|^2 + \text{const.}\end{aligned}}$$

   becomes

$$\frac{\beta_1}{2(1-\alpha_1)(1-\beta_1)} = \frac{\beta_1}{2\beta_1(1-\beta_1)} = \frac{1}{2(1-\beta_1)}.$$

2. This is exactly the coefficient above.
3. Thus, the reconstruction term at $t = 1$ and the consistency terms for $t \geq 2$ all have the same mathematical structure.
4. They can be combined into a single noise-prediction objective.

The final objective function

Combining all terms gives a weighted noise-prediction objective

$$\mathcal{L}(\mathsf{w}) \stackrel{c}{=} -\sum_{t=1}^{T} \mathbb{E}_{\boldsymbol{\epsilon}_t}\left[\frac{\beta_t}{2(1-\alpha_t)(1-\beta_t)} \times \left\|\mathsf{g}\left(\sqrt{\alpha_t}\mathsf{x}+\sqrt{1-\alpha_t}\boldsymbol{\epsilon}_t, \mathsf{w}, t\right)-\boldsymbol{\epsilon}_t\right\|^2\right]$$

1. It was found empirically that training performance could be improved by removing the time-dependent weight.

$$\boxed{\mathcal{L}_{\text{simple}}(\mathsf{w}) = -\sum_{t=1}^{T} \mathbb{E}_{\boldsymbol{\epsilon}_t}\left[\left\|\mathsf{g}\left(\sqrt{\alpha_t}\mathsf{x}+\sqrt{1-\alpha_t}\boldsymbol{\epsilon}_t, \mathsf{w}, t\right)-\boldsymbol{\epsilon}_t\right\|^2\right].}$$

2. Equivalently, we can minimize

$$\boxed{\mathcal{J}_{\text{simple}}(\mathsf{w}) = \mathbb{E}_{\mathsf{x},t,\boldsymbol{\epsilon}}\left[\left\|\mathsf{g}\left(\sqrt{\alpha_t}\mathsf{x}+\sqrt{1-\alpha_t}\boldsymbol{\epsilon}, \mathsf{w}, t\right)-\boldsymbol{\epsilon}\right\|^2\right].}$$

3. t is sampled uniformly from $\{1, \ldots, T\}$.
4. The expectation over uniformly sampled t corresponds to the sum, up to a factor T.

The training algorithm

Input: Training data $\mathcal{D} = \{\mathbf{x}_n\}$

Noise schedule $\{\beta_1, \ldots, \beta_T\}$

Output: Network parameters $\mathbf{w}$

---

for $t \in \{1, \ldots, T\}$ **do**

  $\alpha_t \leftarrow \prod_{\tau=1}^{t}(1 - \beta_\tau)$ // Calculate alphas from betas

end for

repeat

  $\mathbf{x} \sim \mathcal{D}$ // Sample a data point

  $t \sim \text{Uniform}(\{1, \ldots, T\})$ // Sample a diffusion timestep

  $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ // Sample a noise vector

  $\mathbf{z}_t \leftarrow \sqrt{\alpha_t}\mathbf{x} + \sqrt{1 - \alpha_t}\epsilon$ // Evaluate noisy latent variable

  $\mathcal{L}(\mathbf{w}) \leftarrow \|\mathbf{g}(\mathbf{z}_t, \mathbf{w}, t) - \epsilon\|^2$ // Compute loss term

  Take optimizer step

until converged

return $\mathbf{w}$

Notes

1. Do not need to simulate the full forward diffusion chain during training. The closed-form diffusion kernel allows us to sample z_t directly at any time step.
2. The target noise is known exactly. We generated ϵ ourselves, so the network can be trained with supervised regression.
3. The same neural network handles all noise levels. The time step t is supplied as an input, allowing the network to adapt its predictions to different noise intensities.

Generating new samples

1. The learned reverse transition is
$$p(\mathsf{z}_{t-1} \mid \mathsf{z}_t, \mathsf{w}) = \mathcal{N}\left(\boldsymbol{\mu}(\mathsf{z}_t, \mathsf{w}, t), \beta_t \mathsf{I}\right).$$
2. To sample from this Gaussian, use the standard reparameterization
$$\mathsf{z}_{t-1} = \boldsymbol{\mu}(\mathsf{z}_t, \mathsf{w}, t) + \sqrt{\beta_t}\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathsf{I})$$
3. The mean is
$$\boxed{\boldsymbol{\mu}(\mathsf{z}_t, \mathsf{w}, t) = \frac{1}{\sqrt{1-\beta_t}}\left[\mathsf{z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}}\mathsf{g}(\mathsf{z}_t, \mathsf{w}, t)\right].}$$
4. Substituting into above equation
$$\boxed{\mathsf{z}_{t-1} = \frac{1}{\sqrt{1-\beta_t}}\left[\mathsf{z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}}\mathsf{g}(\mathsf{z}_t, \mathsf{w}, t)\right] + \sqrt{\beta_t}\boldsymbol{\epsilon}.}$$
5. Denoising: Estimate and subtract a fraction of the noise using the trained network.
6. Stochastic sampling: Add fresh Gaussian noise to sample from the reverse conditional distribution.

The sampling algorithm

Input: Trained denoising network $\mathbf{g}(\mathbf{z}, \mathbf{w}, t)$

Noise schedule $\{\beta_1, \ldots, \beta_T\}$

Output: Sample vector $\mathbf{x}$ in data space

---

for $t \in \{1, \ldots, T\}$ **do**

$\alpha_t \leftarrow \prod_{\tau=1}^{t}(1-\beta_\tau)$ // Calculate alphas from betas

end for

$\mathbf{z}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ // Sample from final latent space

for $t \in \{T, \ldots, 2\}$ **do**

// Evaluate network output

$$\boldsymbol{\mu}(\mathbf{z}_t, \mathbf{w}, t) \leftarrow \frac{1}{\sqrt{1-\beta_t}}\left\{\mathbf{z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}}\mathbf{g}(\mathbf{z}_t, \mathbf{w}, t)\right\}$$

$\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ // Sample a noise vector

$\mathbf{z}_{t-1} \leftarrow \boldsymbol{\mu}(\mathbf{z}_t, \mathbf{w}, t) + \sqrt{\beta_t}\boldsymbol{\epsilon}$ // Add scaled noise

end for

$$\mathbf{x} = \frac{1}{\sqrt{1-\beta_1}}\left\{\mathbf{z}_1 - \frac{\beta_1}{\sqrt{1-\alpha_1}}\mathbf{g}(\mathbf{z}_1, \mathbf{w}, 1)\right\}$$ // Final denoising step

return $\mathbf{x}$

Why subtract only a fraction of the predicted noise?

1. The network $g(z_t, w, t)$ predicts the total effective noise added to the original data x to produce z_t.
2. When generating z_{t-1}, we only want to move backward by one step.
3. The reverse mean is

$$\boldsymbol{\mu} = \frac{1}{\sqrt{1-\beta_t}} \left[z_t - \boxed{\frac{\beta_t}{\sqrt{1-\alpha_t}} g(z_t, w, t)} \right].$$

4. Why this coefficient? Recall that

$$z_t = \sqrt{\alpha_t} x + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_t.$$

5. The factor $\sqrt{1-\alpha_t}$ describes the cumulative noise scale at time t
6. β_t describes the variance of the noise introduced in a single forward step.
7. The coefficient

$$\frac{\beta_t}{\sqrt{1-\alpha_t}}$$

arises from the exact Gaussian conditional mean $\mathbb{E}_q[z_{t-1} \mid z_t, x]$.

Why adding Gaussian noise during generation?

1. The reverse process is a probabilistic sampling procedure, not merely a deterministic noise-removal algorithm.
2. The reverse transition is

$$p(\mathsf{z}_{t-1} \mid \mathsf{z}_t, \mathsf{w}) = \mathcal{N}(\boldsymbol{\mu}_t, \beta_t \mathsf{I}).$$

3. The network determines the conditional mean $\boldsymbol{\mu}_t$, but the conditional distribution has nonzero variance. Therefore, sampling requires

$$\mathsf{z}_{t-1} = \boldsymbol{\mu}_t + \sqrt{\beta_t}\boldsymbol{\epsilon}.$$

4. If we omitted the noise term, we would obtain

$$\mathsf{z}_{t-1} = \boldsymbol{\mu}_t,$$

which is the conditional mean rather than a sample from the modeled Gaussian distribution.

Why is no noise added at the final step?

1. At the final step $t = 1$,

$$\epsilon = 0.$$

   Therefore,

$$\boxed{\mathsf{x} = \boldsymbol{\mu}(\mathsf{z}_1, \mathsf{w}, 1).}$$

2. We want the final result to be a clean data sample, rather than adding another increment of Gaussian noise after the last denoising operation.
3. If we literally sampled from the Gaussian observation model

$$p(\mathsf{x} \mid \mathsf{z}_1, \mathsf{w}) = \mathcal{N}(\boldsymbol{\mu}_1, \beta_1 \mathsf{I}),$$

   we would add noise even at the last step.

Training versus sampling

Aspect	**Training**	**Sampling**
Starting point	Real data x	Gaussian noise z_T
Time step	Randomly sample t	Traverse $T, T-1, \ldots, 1$
Noisy input	Generated directly using the diffusion kernel	Obtained from previous reverse step
Network output	Predicted noise	Predicted noise
Objective	Minimize noise-prediction MSE	Generate a data sample
Gradients	Required	Not required
Number of network evaluations	One per sampled training example	Approximately T per generated example

1. Training does not require simulating the entire diffusion chain
2. Generation requires sequential reverse steps because each z_{t-1} depends on the previously generated z_t

Discrete-time and continuous-time Gaussian diffusion

1. Discrete-time Gaussian diffusion uses Markov transitions:

$$q(z_t \mid z_{t-1}) = \mathcal{N}(\sqrt{1-\beta_t}\, z_{t-1}, \beta_t I).$$

2. Continuous-time Gaussian diffusion uses an SDE:

$$dz_t = f(z_t, t)\, dt + g(t)\, dW_t.$$

3. The former has finitely many corruption steps; the latter specifies dynamics at every time.
4. Suitable time discretizations of SDEs yield discrete-time Gaussian transitions.
5. Both use continuous-valued states; neither formulation alone determines whether a process is deterministic.

Deterministic and stochastic continuous-time dynamics

1. Flow matching commonly learns an ODE:

$$\frac{dx_t}{dt} = v_\theta(x_t, t).$$

2. Diffusion models commonly specify an SDE:

$$dz_t = f(z_t, t)\, dt + g(t)\, dW_t.$$

3. The associated probability flow ODE is

$$\frac{dz_t}{dt} = f(z_t, t) - \frac{g^2(t)}{2} \nabla_z \log p_t(z_t).$$

4. Under suitable conditions, the SDE and probability flow ODE have identical one-time marginal distributions, but different sample trajectories.
5. This equivalence concerns probability evolution, not equality of paths.

Why discrete diffusion for language?

1. A text sequence is $x = (x_1, \ldots, x_L)$ with $x_i \in \mathcal{V}$, where $\mathcal{V}$ is a finite vocabulary.
2. Gaussian perturbations are natural for real-valued data, but tokens are categorical symbols.
3. Discrete diffusion instead corrupts tokens through categorical transitions.
4. A learned reverse process gradually reconstructs a clean sequence from a simple noise distribution.
5. The underlying time variable can be discrete or continuous; the state remains discrete in either case.

Categorical corruption transitions

1. For $K = |\mathcal{V}|$, a transition matrix Q_t specifies

$$[Q_t]_{ij} = q(z_t = j \mid z_{t-1} = i), \qquad \sum_j [Q_t]_{ij} = 1.$$

2. Uniform replacement is one example:

$$Q_t = (1 - \beta_t) I + \beta_t \frac{1 1^\top}{K}.$$

3. At each step a token is retained with probability $1 - \beta_t$, or replaced by a uniform draw with probability β_t.
4. For independent corruption across positions,

$$q(z_t \mid z_{t-1}) = \prod_{i=1}^{L} q(z_{t,i} \mid z_{t-1,i}).$$

5. Independence of the forward corruption does not imply independence of tokens in the learned data distribution.

Cumulative categorical corruption

1. The forward Markov chain starts from $z_0 = x$:

$$q(z_{1:T} \mid x) = \prod_{t=1}^{T} q(z_t \mid z_{t-1}).$$

2. Define the cumulative transition matrix

$$\overline{Q}_t = Q_1 Q_2 \cdots Q_t.$$

3. For a clean token $x = i$,

$$q(z_t = j \mid x = i) = \mathrm{e}_i^\top \overline{Q}_t \mathrm{e}_j.$$

4. Thus z_t can be sampled directly from x without simulating every earlier corruption step.
5. This is the categorical counterpart of direct noising in Gaussian diffusion.

Why direct sampling from the clean data matters

1. Training repeatedly draws a clean example x, a time t, and a noisy sample z_t.
2. Simulating $z_1, \ldots, z_t$ would make training cost grow with the selected noise level.
3. The cumulative kernel $q(z_t \mid x)$ avoids this sequential simulation.
4. For uniform replacement, let $\overline{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$. Then

$$\overline{Q}_t = \overline{\alpha}_t I + (1-\overline{\alpha}_t)\frac{11^\top}{K}.$$

5. Hence each position can be sampled by retaining its clean token with probability $\overline{\alpha}_t$, or resampling uniformly otherwise.

Absorbing-mask diffusion

1. Add a special symbol $m = [\mathrm{MASK}]$ to the vocabulary.
2. At step t, an unmasked token is kept with probability $1 - \beta_t$ and replaced by m with probability β_t.
3. Once masked, a token remains masked in the forward process:

$$q(z_t = m \mid z_{t-1} = m) = 1.$$

4. Define the survival probability

$$\alpha_t = \prod_{s=1}^{t} (1 - \beta_s), \qquad \alpha_0 = 1.$$

5. For a clean token $x \neq m$, the marginal law is

$$q(z_t \mid x) = \alpha_t 1[z_t = x] + (1 - \alpha_t) 1[z_t = m].$$

Interpreting the masking schedule

1. α_t is the probability that a clean token has survived all corruption steps through time t.
2. The probability of being masked by time t is $1 - \alpha_t$.
3. A schedule with $\alpha_T \approx 0$ produces an almost fully masked terminal sequence; $\alpha_T = 0$ yields a fully masked terminal sequence.
4. For independently corrupted positions,

$$q(z_t \mid x) = \prod_{i=1}^{L} \left(\alpha_t 1[z_{t,i} = x_i] + (1 - \alpha_t) 1[z_{t,i} = m]\right).$$

5. Although the forward process factorizes by position, the reverse predictor can use the entire corrupted sequence as context.

Reverse transitions: what must be learned?

1. Generation reverses the corruption process:

$$p_\theta(z_{t-1} \mid z_t), \qquad t = T, T-1, \ldots, 1.$$

2. The exact forward posterior $q(z_{t-1} \mid z_t, x)$ is tractable when the clean token x is known.
3. During generation x is unknown, so a network predicts clean-token probabilities $p_\theta(x_i \mid z_t, t)$.
4. The predicted clean-token distribution can be combined with the exact forward posterior to construct a reverse transition.
5. This separates known corruption mathematics from the learned statistical structure of language.

Deriving the absorbing-mask reverse posterior

1. Condition on clean token $x \neq m$ and observe $z_t = m$.
2. The event $z_{t-1} = x,\ z_t = m$ has probability $\alpha_{t-1}\beta_t$.
3. Since $\alpha_t = \alpha_{t-1}(1 - \beta_t)$,

$$\alpha_{t-1}\beta_t = \alpha_{t-1} - \alpha_t.$$

4. The event $z_t = m$ has probability $1 - \alpha_t$.
5. Bayes' rule gives

$$\boxed{q(z_{t-1} = x \mid z_t = m, x) = \frac{\alpha_{t-1} - \alpha_t}{1 - \alpha_t}.}$$

Completing the reverse posterior

1. If $z_t = m$, the token was already masked at time $t-1$, or became masked during step t.
2. The probability it was already masked is

$$\boxed{q(z_{t-1} = m \mid z_t = m, x) = \frac{1 - \alpha_{t-1}}{1 - \alpha_t}.}$$

3. The two posterior probabilities sum to one.
4. If $z_t \neq m$, the forward absorbing process implies $z_{t-1} = z_t = x$ with probability one.
5. Reverse sampling therefore only needs to decide whether to reveal a currently masked token, and which clean token to reveal.

Constructing a learned reverse transition

1. Let $\pi_t = \dfrac{\alpha_{t-1} - \alpha_t}{1 - \alpha_t}$ be the probability of revealing a masked position in the exact posterior.
2. For $z_{t,i} = m$, a simple learned reverse kernel is

$$p_\theta(z_{t-1,i} = a \mid z_t) = \pi_t \, p_\theta(x_i = a \mid z_t, t), \quad a \in \mathcal{V},$$
$$p_\theta(z_{t-1,i} = m \mid z_t) = 1 - \pi_t.$$

3. For $z_{t,i} \neq m$, keep $z_{t-1,i} = z_{t,i}$.
4. The network can predict all positions jointly through shared bidirectional context, even if a sampler factorizes the final positionwise draws.

Training diffusion language models

1. Draw a clean sequence $x \sim p_{\text{data}}$, a time t, and $z_t \sim q(z_t \mid x)$.
2. Train a network to predict the original token at masked positions:

$$\mathcal{L}_{\text{mask}}(\theta) = \mathbb{E}\left[-\sum_{i=1}^{L} 1[z_{t,i} = m] \log p_\theta(x_i \mid z_t, t)\right].$$

3. Because z_t includes both left and right context, a bidirectional architecture can use both.
4. Sampling the corruption time across different noise levels teaches the network to denoise under different amounts of missing information.
5. The distribution used to sample t and any explicit time weights affect the effective training objective.

Why cross-entropy learns clean-token probabilities

1. Fix a corrupted context z_t and a masked position i.
2. Let $r(a) = q(x_i = a \mid z_t, t)$ denote the true clean-token posterior.
3. The conditional expected cross-entropy is

$$\begin{aligned}\mathbb{E}[-\log p_\theta(x_i \mid z_t, t) \mid z_t] &= -\sum_{a \in \mathcal{V}} r(a) \log p_\theta(a \mid z_t, t) \\ &= H(r) + D_{\mathrm{KL}}(r \| p_\theta).\end{aligned}$$

4. The entropy $H(r)$ does not depend on θ; the minimum is attained when $p_\theta(\cdot \mid z_t, t) = r(\cdot)$, if the model can represent it.
5. Thus denoising cross-entropy estimates the clean-token posterior needed by the reverse process.

Discrete diffusion as a latent-variable model

1. The reverse generative model introduces latent corrupted sequences $z_{1:T}$:

$$p_\theta(x, z_{1:T}) = p(z_T)\, p_\theta(x \mid z_1) \prod_{t=2}^{T} p_\theta(z_{t-1} \mid z_t).$$

2. The likelihood sums over all latent trajectories:

$$p_\theta(x) = \sum_{z_{1:T}} p_\theta(x, z_{1:T}).$$

3. The forward corruption process $q(z_{1:T} \mid x)$ supplies a tractable variational distribution.
4. Jensen's inequality gives

$$\log p_\theta(x) \geq \mathbb{E}_q \left[\log \frac{p_\theta(x, z_{1:T})}{q(z_{1:T} \mid x)} \right].$$

Decomposing the diffusion ELBO

1. The negative ELBO can be organized as

$$-\mathrm{ELBO}(x) = \underbrace{D_{\mathrm{KL}}(q(z_T \mid x) \| p(z_T))}_{\text{Terminal prior}} + \sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(z_t|x)} D_{\mathrm{KL}}(q(z_{t-1} \mid z_t, x) \| p_\theta(z_{t-1} \mid z_t))}_{\text{Reverse-transition matching}} - \underbrace{\mathbb{E}_{q(z_1|x)} \log p_\theta(x \mid z_1)}_{\text{Reconstruction}}.$$

2. This decomposition follows from the Markov factorization and Bayes' rule for the forward posterior.
3. It clarifies why the exact forward posterior is useful for training the reverse model.

Masked-token loss versus the diffusion ELBO

1. The basic masked-token objective is a denoising cross-entropy averaged over corruption times.
2. The ELBO additionally specifies reverse-transition KL terms, a terminal-prior term, and reconstruction.
3. For absorbing-mask diffusion, some reverse KL terms simplify to weighted clean-token prediction losses under suitable parameterizations.
4. The resulting weights depend on the masking schedule and the time-sampling distribution.
5. Therefore an *unweighted* masked-token cross-entropy is generally not numerically identical to the negative ELBO, even when both train a useful denoiser.

Continuous-time diffusion on discrete states

1. A continuous-time Markov chain (CTMC) has discrete states but evolves in continuous time.
2. Let $q_t(i) = \Pr(Z_t = i)$ and let R_t be its rate matrix:

$$[R_t]_{ij} \geq 0 \quad (i \neq j), \qquad \sum_j [R_t]_{ij} = 0.$$

3. The diagonal rate is $[R_t]_{ii} = -\sum_{j \neq i} [R_t]_{ij}$.
4. Over a short interval $h > 0$,

$$\Pr(Z_{t+h} = j \mid Z_t = i) = 1[i = j] + h[R_t]_{ij} + o(h).$$

5. Sample paths remain constant between random jump times; the state never takes fractional token values.

Deriving the master equation

1. Apply the law of total probability over the state at time t:

$$q_{t+h}(j) = \sum_i q_t(i) \Pr(Z_{t+h} = j \mid Z_t = i).$$

2. Substitute the short-time transition expansion:

$$q_{t+h}(j) = q_t(j) + h \sum_i q_t(i)[R_t]_{ij} + o(h).$$

3. Divide by h and let $h \to 0$:

$$\boxed{\frac{dq_t(j)}{dt} = \sum_i q_t(i)[R_t]_{ij}.}$$

4. Row sums of R_t vanish, so $\frac{d}{dt} \sum_j q_t(j) = 0$: total probability is conserved.

Probability flux in the master equation

1. Expand the diagonal term using $[R_t]_{jj} = -\sum_{i \neq j} [R_t]_{ji}$:

$$\frac{dq_t(j)}{dt} = \sum_{i \neq j} \left(q_t(i)[R_t]_{ij} - q_t(j)[R_t]_{ji} \right).$$

2. The first term is incoming probability flow from state i to state j.
3. The second term is outgoing probability flow from state j to state i.
4. This is the discrete-state analogue of the continuity equation:

$$\partial_t p_t(x) + \nabla \cdot \big(p_t(x) v_t(x) \big) = 0.$$

5. Unlike the continuous equation, the master equation uses sums over transitions rather than spatial derivatives.

Discrete time as a numerical approximation to a CTMC

1. For a small time increment h, the CTMC transition matrix satisfies
$$Q_{t,t+h} = I + hR_t + o(h).$$
2. Thus discrete simulation steps can approximate continuous-time jump dynamics.
3. To obtain valid first-order transition probabilities, choose h small enough that $1 + h[R_t]_{ii} \geq 0$ for all states.
4. For a time-homogeneous rate matrix R, the exact finite-time transition matrix is
$$Q_{0,t} = \exp(tR).$$
5. A discrete-time implementation does not make the underlying CTMC a discrete-time mathematical model.

Iterative parallel decoding

1. In absorbing-mask diffusion, generation can begin from $z_T = (m, \ldots, m)$.
2. At each reverse step, the network predicts clean-token distributions at many positions simultaneously:

$$p_\theta(x_i \mid z_t, t), \qquad i = 1, \ldots, L.$$

3. A reverse transition reveals selected masked tokens, possibly at many positions in one round.
4. The sequence is refined through

$$z_T \longrightarrow z_{T-1} \longrightarrow \cdots \longrightarrow z_1 \longrightarrow x.$$

5. Parallel predictions reduce the number of sequential token-generation rounds, but each refinement round still requires neural-network computation.

Autoregressive versus diffusion language models

1. Autoregressive models use the chain-rule factorization

$$p_\theta(x) = \prod_{i=1}^{L} p_\theta(x_i \mid x_1, \ldots, x_{i-1}).$$

2. Diffusion language models instead learn to denoise corrupted sequences across multiple noise levels.
3. Key distinctions:
   - Training: next-token prediction versus denoising objectives.
   - Context: causal attention versus potentially bidirectional attention.
   - Sampling: sequential token generation versus iterative multi-position refinement.
   - Likelihood: direct autoregressive evaluation versus latent-state marginalization or bounds.
4. Neither paradigm is universally faster or more accurate; performance depends on model, sampler, sequence length, and hardware.

Connections to continuous diffusion and flow matching

1. Continuous and discrete diffusion both specify a forward corruption process and learn a reverse generative process.
2. Continuous states are naturally described by Gaussian noise, scores, SDEs, and Fokker–Planck equations.
3. Discrete states are naturally described by categorical kernels, token posteriors, CTMCs, and master equations.
4. Both can be formulated as latent-variable generative models and related to variational learning.
5. Discrete flow matching provides another viewpoint: prescribe a probability path over categorical states and learn jump rates that realize it.
6. The common principle is probability transport; the operators governing that transport depend on the state space.

Outline

1 Introduction

2 Flow Matching

3 Diffusion Models

4 Score Matching
- Score Functions and Score Matching
- Theoretical Analysis
- Langevin Dynamics and Sampling
- Conditional Guidance

5 Connections

6 Frontiers

7 Latent Variable Models

8 Variational Inference

The score function

1. The score function is defined as

$$\boxed{\mathrm{s}(\mathrm{x}) = \nabla_{\mathrm{x}} \ln p(\mathrm{x})}$$

   where $\mathrm{x} \in \mathbb{R}^D$ and $p(\mathrm{x})$ is a probability density.

2. The score is a vector with the same dimensionality as x:

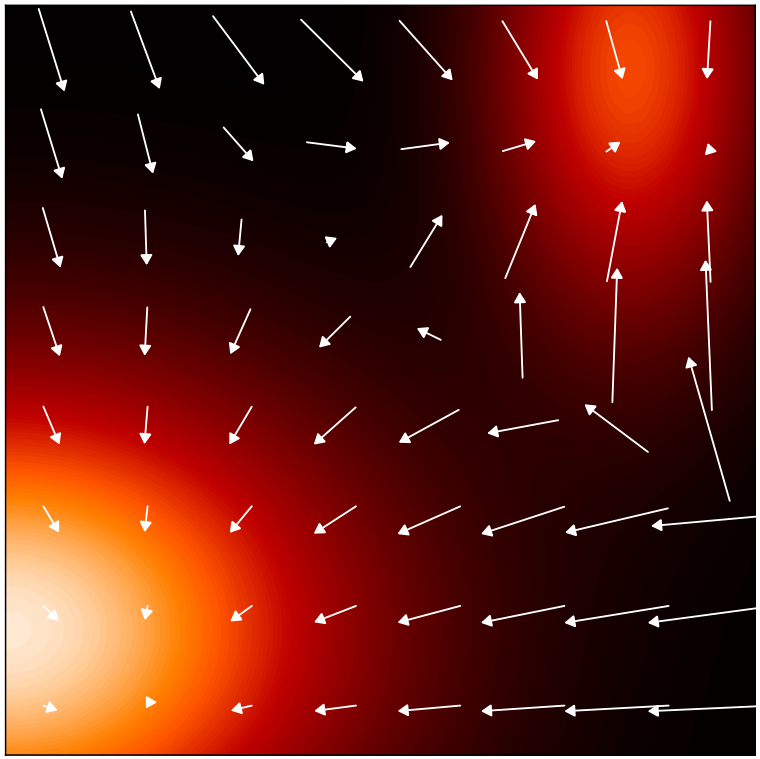
$$\mathrm{s}(\mathrm{x}) = \begin{pmatrix} \frac{\partial \ln p(\mathrm{x})}{\partial x_1} \\ \vdots \\ \frac{\partial \ln p(\mathrm{x})}{\partial x_D} \end{pmatrix}.$$

3. The gradient is taken with respect to the data, not the parameters of the distribution.

Example: Gaussian distribution

1. Suppose
$$p(\mathrm{x}) = \mathcal{N}(\mathrm{x} \mid \boldsymbol{\mu}, \sigma^2 \mathrm{I}).$$
2. Taking the logarithm gives
$$\ln p(\mathrm{x}) = -\frac{D}{2} \ln(2\pi\sigma^2) - \frac{1}{2\sigma^2} \|\mathrm{x} - \boldsymbol{\mu}\|^2.$$
3. Differentiating with respect to x,
$$\boxed{\mathrm{s}(\mathrm{x}) = -\frac{\mathrm{x} - \boldsymbol{\mu}}{\sigma^2} = \frac{\boldsymbol{\mu} - \mathrm{x}}{\sigma^2}.}$$
4. This has an intuitive interpretation:
   - $\boldsymbol{\mu} - \mathrm{x}$ is the vector pointing from x toward $\boldsymbol{\mu}$.
   - The score points toward the mean $\boldsymbol{\mu}$, where the density is highest.
   - The farther x is from the mean, the larger the score magnitude.
   - The score is zero at the mean.
   - The score defines a vector field pointing in the direction of increasing log probability.

Mixture of Gaussian



1. The figure shows a mixture of Gaussian distributions.
2. The bright regions indicate high probability density, and the arrows represent the score vectors.
3. The arrows generally point toward nearby high-density regions, although they can become small or change direction near stationary points.

Why learning the score is sufficient

1. Considers two densities $p(\mathrm{x})$ and $q(\mathrm{x})$ whose scores are equal:

$$\nabla_{\mathrm{x}} \ln p(\mathrm{x}) = \nabla_{\mathrm{x}} \ln q(\mathrm{x}).$$

   Rearranging,

$$\nabla_{\mathrm{x}} [\ln q(\mathrm{x}) - \ln p(\mathrm{x})] = 0.$$

2. Therefore,

$$\ln q(\mathrm{x}) - \ln p(\mathrm{x}) = C.$$

   Exponentiating,

$$\boxed{q(\mathrm{x}) = Kp(\mathrm{x})}$$

   where K is a constant.

3. If both densities are normalized over the same domain, then $K = 1$.
4. The score determines the probability density up to a multiplicative constant.

Energy-based models

1. This is particularly useful for energy-based models.
2. Suppose
$$p(\mathrm{x}) = \frac{1}{Z} \exp[-E(\mathrm{x})].$$
Then
$$\begin{aligned} \nabla_{\mathrm{x}} \ln p(\mathrm{x}) &= \nabla_{\mathrm{x}}[-E(\mathrm{x}) - \ln Z] \\ &= -\nabla_{\mathrm{x}} E(\mathrm{x}). \end{aligned}$$
Hence,
$$\boxed{\mathrm{s}(\mathrm{x}) = -\nabla_{\mathrm{x}} E(\mathrm{x}).}$$
3. The normalization constant Z, which may be extremely difficult to compute, disappears.
4. This is one of the major advantages of score-based modeling.

Score matching loss

1. We introduce a neural network

$$s(x, w)$$

to approximate the true score

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

2. The natural training objective is the expected squared error:

$$\boxed{J(w) = \frac{1}{2} \int \|s(x, w) - \nabla_x \ln p(x)\|^2 \, p(x) \, dx.}$$

3. Equivalently,

$$J(w) = \frac{1}{2} \mathbb{E}_{x \sim p} \left[\|s(x, w) - \nabla_x \ln p(x)\|^2 \right].$$

4. The objective is minimized when

$$s(x, w) = \nabla_x \ln p(x)$$

almost everywhere under p, assuming the model is sufficiently expressive.

Two ways to represent the score

1. Approach A: Direct vector prediction. Use a neural network

$$s(x, w) : \mathbb{R}^D \to \mathbb{R}^D.$$

   Its output has the same dimensionality as its input.

2. Approach B: Gradient of a scalar function. Use a scalar network

$$\phi(x, w) : \mathbb{R}^D \to \mathbb{R}$$

   and define

$$s(x, w) = \nabla_x \phi(x, w).$$

3. The second approach guarantees that the predicted score is a gradient field, but it requires differentiating through an input gradient during training, making it more computationally expensive.
4. The first approach is more flexible and is commonly used in diffusion models.

Why the original score loss is difficult to minimize

1. We do not know $p(\mathrm{x})$.
2. We only have a finite training dataset

$$\mathcal{D} = \{\mathrm{x}_1, \ldots, \mathrm{x}_N\}.$$

3. The empirical distribution is

$$\boxed{p_{\mathcal{D}}(\mathrm{x}) = \frac{1}{N} \sum_{n=1}^{N} \delta(\mathrm{x} - \mathrm{x}_n).}$$

   Here δ is the Dirac delta distribution (next page).
4. This empirical distribution concentrates all probability mass at the training samples.
5. It does not have an ordinary, differentiable log density, so its score cannot be computed in the usual way.

Review: Dirac delta distribution

1. The Dirac delta distribution $\delta(x-a)$ is defined by

$$\delta(x-a) = 0, \qquad x \neq a,$$

$$\int_{-\infty}^{\infty} \delta(x-a)\,dx = 1,$$

$$\int_{-\infty}^{\infty} f(x)\delta(x-a)\,dx = f(a).$$

2. Sampling from $\delta(x-a)$ always returns a (i.e., it is deterministic).
3. For example, we need the below result on the next page

$$\begin{aligned} q_\sigma(\mathrm{z}) &= \int q(\mathrm{z} \mid \mathrm{x}, \sigma) \frac{1}{N} \sum_{n=1}^{N} \delta(\mathrm{x} - \mathrm{x}_n)\,d\mathrm{x} \\ &= \frac{1}{N} \sum_{n=1}^{N} \int q(\mathrm{z} \mid \mathrm{x}, \sigma)\delta(\mathrm{x} - \mathrm{x}_n)\,d\mathrm{x} \\ &= \frac{1}{N} \sum_{n=1}^{N} q(\mathrm{z} \mid \mathrm{x}_n, \sigma). \end{aligned}$$

Gaussian smoothing and the Parzen density

1. Introduce a conditional noise distribution $q(\mathrm{z} \mid \mathrm{x}, \sigma)$.
2. The smoothed density is defined as

$$\boxed{q_\sigma(\mathrm{z}) = \int q(\mathrm{z} \mid \mathrm{x}, \sigma) p(\mathrm{x})\, d\mathrm{x}.}$$

3. For Gaussian noise, $q(\mathrm{z} \mid \mathrm{x}, \sigma) = \mathcal{N}(\mathrm{z} \mid \mathrm{x}, \sigma^2 \mathrm{I})$.
4. Substituting the empirical distribution gives

$$q_\sigma(\mathrm{z}) = \int q(\mathrm{z} \mid \mathrm{x}, \sigma) \frac{1}{N} \sum_{n=1}^{N} \delta(\mathrm{x} - \mathrm{x}_n)\, d\mathrm{x} = \frac{1}{N} \sum_{n=1}^{N} q(\mathrm{z} \mid \mathrm{x}_n, \sigma).$$

$$\boxed{q_\sigma(\mathrm{z}) = \frac{1}{N} \sum_{n=1}^{N} \mathcal{N}(\mathrm{z} \mid \mathrm{x}_n, \sigma^2 \mathrm{I}).}$$

5. It has a smooth density for every $\sigma > 0$.

Modified score loss

1. Train the network to approximate the score of the smoothed distribution:

$$\boxed{J(\mathsf{w}) = \frac{1}{2} \int \|\mathsf{s}(\mathsf{z}, \mathsf{w}) - \nabla_{\mathsf{z}} \ln q_\sigma(\mathsf{z})\|^2 \, q_\sigma(\mathsf{z}) \, d\mathsf{z}.}$$

2. However, the score

$$\nabla_{\mathsf{z}} \ln q_\sigma(\mathsf{z})$$

still appears difficult to compute, since q_σ is a mixture over all training samples.

3. The key insight is that we can replace the score of the marginal distribution with the score of the conditional noise distribution inside a suitable expected squared-error objective.

Equivalent objective

1. We show that minimizing

$$\boxed{J_1(\mathrm{w}) = \frac{1}{2} \int \left\| \mathrm{s}(\mathrm{z}, \mathrm{w}) - \nabla_{\mathrm{z}} \ln q_\sigma(\mathrm{z}) \right\|^2 q_\sigma(\mathrm{z})\, d\mathrm{z}}$$

   is equivalent, up to an additive constant, to minimizing

$$\boxed{J_2(\mathrm{w}) = \frac{1}{2} \iint \left\| \mathrm{s}(\mathrm{z}, \mathrm{w}) - \nabla_{\mathrm{z}} \ln q(\mathrm{z} \mid \mathrm{x}, \sigma) \right\|^2 q(\mathrm{z} \mid \mathrm{x}, \sigma) p(\mathrm{x})\, d\mathrm{z}\, d\mathrm{x}.}$$

2. Why this matters: Replace an unknown marginal score with a conditional score that can be computed analytically.

Equivalence of $J_1(w)$ and $J_2(w)$

1. For brevity, write
$$s_w(z) = s(z, w).$$
2. Expanding the squared norm,
$$\begin{aligned} J_1(w) = \frac{1}{2} \int \|s_w(z)\|^2 q_\sigma(z)\, dz \\ - \int s_w(z)^\top \nabla_z \ln q_\sigma(z) q_\sigma(z)\, dz \\ + \frac{1}{2} \int \|\nabla_z \ln q_\sigma(z)\|^2 q_\sigma(z)\, dz. \end{aligned}$$
3. The final term does not depend on w, so it can be absorbed into a constant.

Equivalence of $J_1(w)$ and $J_2(w)$

1. Use the identity
$$\nabla_z \ln q_\sigma(z) q_\sigma(z) = \nabla_z q_\sigma(z).$$
Therefore,
$$\int s_w(z)^\top \nabla_z \ln q_\sigma(z) q_\sigma(z)\, dz$$
$$= \int s_w(z)^\top \nabla_z q_\sigma(z)\, dz.$$
2. Now substitute
$$q_\sigma(z) = \int q(z \mid x, \sigma) p(x)\, dx.$$
3. Assuming differentiation and integration can be interchanged,
$$\nabla_z q_\sigma(z) = \int \nabla_z q(z \mid x, \sigma) p(x)\, dx.$$

Equivalence of $J_1(w)$ and $J_2(w)$

1. Thus,

$$\begin{aligned}
&\int s_w(z)^\top \nabla_z q_\sigma(z)\, dz \\
&= \iint s_w(z)^\top \nabla_z q(z \mid x, \sigma) p(x)\, dx\, dz \\
&= \iint s_w(z)^\top \nabla_z \ln q(z \mid x, \sigma) q(z \mid x, \sigma) p(x)\, dx\, dz.
\end{aligned}$$

2. This is exactly the cross term obtained by expanding J_2.

Equivalence of $J_1(w)$ and $J_2(w)$

1. The first term of J_1 is
$$\frac{1}{2}\int \|s_w(z)\|^2 q_\sigma(z)\, dz.$$
2. Substituting the definition of q_σ,
$$\frac{1}{2}\iint \|s_w(z)\|^2 q(z \mid x, \sigma) p(x)\, dx\, dz.$$
3. This is also exactly the first quadratic term of J_2.
4. Hence,

$$\boxed{J_1(w) = \frac{1}{2}\iint \|s(z,w) - \nabla_z \ln q(z \mid x, \sigma)\|^2 \cdot q(z \mid x, \sigma) p(x)\, dz\, dx + \text{const.}}$$

The score matching objective

1. Now substitute the empirical distribution into $J_2(\mathsf{w})$

$$p_{\mathcal{D}}(\mathsf{x}) = \frac{1}{N}\sum_{n=1}^{N}\delta(\mathsf{x}-\mathsf{x}_n)$$

$$\boxed{J_2(\mathsf{w}) = \frac{1}{2}\iint \|\mathsf{s}(\mathsf{z},\mathsf{w}) - \nabla_{\mathsf{z}} \ln q(\mathsf{z}\mid \mathsf{x},\sigma)\|^2 \, q(\mathsf{z}\mid \mathsf{x},\sigma)p(\mathsf{x})\,d\mathsf{z}\,d\mathsf{x}.}$$

2. We obtain

$$J_2(\mathsf{w}) = \frac{1}{2N}\sum_{n=1}^{N}\int \|\mathsf{s}(\mathsf{z},\mathsf{w}) - \nabla_{\mathsf{z}} \ln q(\mathsf{z}\mid \mathsf{x}_n,\sigma)\|^2 \, q(\mathsf{z}\mid \mathsf{x}_n,\sigma)\,d\mathsf{z}$$

$$= \boxed{\frac{1}{2N}\sum_{n=1}^{N}\mathbb{E}_{\mathsf{z}\sim q(\mathsf{z}|\mathsf{x}_n,\sigma)}\left[\|\mathsf{s}(\mathsf{z},\mathsf{w}) - \nabla_{\mathsf{z}} \ln q(\mathsf{z}\mid \mathsf{x}_n,\sigma)\|^2\right]}$$

$$= \frac{1}{2}\mathbb{E}_{(\mathsf{x},\mathsf{z})\sim p(\mathsf{x},\mathsf{z})}\left[\|\mathsf{s}(\mathsf{z},\mathsf{w}) - \nabla_{\mathsf{z}} \ln q(\mathsf{z}\mid \mathsf{x},\sigma)\|^2\right],$$

which can be optimized by sampling training data and adding noise.

Gaussian score matching

1. Recall that the Gaussian Parzen kernel is

$$q(\mathsf{z} \mid \mathsf{x}, \sigma) = \mathcal{N}(\mathsf{z} \mid \mathsf{x}, \sigma^2 \mathsf{I}).$$

2. We can generate a noisy sample by

$$\mathsf{z} = \mathsf{x} + \sigma \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathsf{I}).$$

3. The log density is

$$\ln q(\mathsf{z} \mid \mathsf{x}, \sigma) = -\frac{D}{2} \ln(2\pi\sigma^2) - \frac{\|\mathsf{z} - \mathsf{x}\|^2}{2\sigma^2}.$$

4. Taking the gradient with respect to z,

$$\nabla_{\mathsf{z}} \ln q(\mathsf{z} \mid \mathsf{x}, \sigma) = -\frac{\mathsf{z} - \mathsf{x}}{\sigma^2} = -\frac{\sigma\boldsymbol{\epsilon}}{\sigma^2}.$$

5. Therefore,

$$\boxed{\nabla_{\mathsf{z}} \ln q(\mathsf{z} \mid \mathsf{x}, \sigma) = -\frac{\boldsymbol{\epsilon}}{\sigma}.}$$

Denoising score matching: training algorithm

Input: Dataset $\mathcal{D} = \{\mathsf{x}_n\}_{n=1}^N$, noise level σ, learning rate η.
Output: Trained score network $\mathsf{s}(\mathsf{z}, \mathsf{w})$.

1. Initialize network parameters w.
2. Repeat until convergence:
   1. Sample a mini-batch $\{\mathsf{x}_i\}_{i=1}^B$ from $\mathcal{D}$.
   2. Sample noise and construct noisy data:

$$\boldsymbol{\epsilon}_i \sim \mathcal{N}(0, \mathsf{I}), \qquad \mathsf{z}_i = \mathsf{x}_i + \sigma \boldsymbol{\epsilon}_i.$$

   3. Compute the target scores:

$$\nabla_{\mathsf{z}_i} \ln q(\mathsf{z}_i \mid \mathsf{x}_i, \sigma) = -\frac{\boldsymbol{\epsilon}_i}{\sigma}.$$

   4. Compute the mini-batch loss:

$$\boxed{\mathcal{L}(\mathsf{w}) = \frac{1}{2B} \sum_{i=1}^{B} \left\| \mathsf{s}(\mathsf{z}_i, \mathsf{w}) + \frac{\boldsymbol{\epsilon}_i}{\sigma} \right\|^2}$$

   5. Update parameters:

$$\mathsf{w} \leftarrow \mathsf{w} - \eta \nabla_{\mathsf{w}} \mathcal{L}(\mathsf{w}).$$

Why does denoising score matching work?

1. Recall the denoising score matching objective:

$$J_2(\mathsf{w}) = \frac{1}{2}\mathbb{E}_{\mathsf{x},\mathsf{z}}\left[\left\|\mathsf{s}(\mathsf{z},\mathsf{w}) - \nabla_{\mathsf{z}} \ln q(\mathsf{z} \mid \mathsf{x}, \sigma)\right\|^2\right].$$

2. By the theory of squared-error regression, the optimal predictor is the conditional expectation of the target given the input:

$$\boxed{\mathsf{s}^*(\mathsf{z}) = \mathbb{E}_{\mathsf{x}|\mathsf{z}}\left[\nabla_{\mathsf{z}} \ln q(\mathsf{z} \mid \mathsf{x}, \sigma)\right].}$$

3. We will show that this conditional expectation equals the score of the noisy marginal distribution:

$$\boxed{\mathsf{s}^*(\mathsf{z}) = \nabla_{\mathsf{z}} \ln q_\sigma(\mathsf{z}),}$$

where

$$q_\sigma(\mathsf{z}) = \int q(\mathsf{z} \mid \mathsf{x}, \sigma) p_{\mathcal{D}}(\mathsf{x})\, d\mathsf{x}.$$

Why does denoising score matching work? (Proof)

1. Express the conditional expectation as an integral:

$$\begin{aligned} s^*(z) &= \mathbb{E}_{x|z}\left[\nabla_z \ln q(z \mid x, \sigma)\right] \\ &= \int \nabla_z \ln q(z \mid x, \sigma) p(x \mid z)\, dx. \end{aligned}$$

2. Apply Bayes' rule:

$$p(x \mid z) = \frac{q(z \mid x, \sigma) p_{\mathcal{D}}(x)}{q_\sigma(z)}.$$

3. Substitute and simplify:

$$\begin{aligned} s^*(z) &= \frac{1}{q_\sigma(z)} \int p_{\mathcal{D}}(x) \nabla_z q(z \mid x, \sigma)\, dx \\ &= \frac{1}{q_\sigma(z)} \nabla_z \int p_{\mathcal{D}}(x) q(z \mid x, \sigma)\, dx \\ &= \frac{\nabla_z q_\sigma(z)}{q_\sigma(z)} \\ &= \boxed{\nabla_z \ln q_\sigma(z)}. \end{aligned}$$

Denoising score matching: key takeaways

1. The marginal score $\nabla_z \ln q_\sigma(z)$ is generally difficult to compute directly.
2. The conditional score $\nabla_z \ln q(z \mid x, \sigma)$ is analytically available for Gaussian noise:
$$\nabla_z \ln q(z \mid x, \sigma) = -\frac{z - x}{\sigma^2}.$$
3. Minimizing $J_2(w)$ learns the conditional expectation of these known scores:
$$\boxed{\begin{aligned} s^*(z) &= \mathbb{E}_{x|z}\left[\nabla_z \ln q(z \mid x, \sigma)\right] \\ &= \nabla_z \ln q_\sigma(z). \end{aligned}}$$
4. Thus, we can learn the marginal score without explicitly evaluating the marginal density $q_\sigma(z)$ or its gradient.

Connection to flow matching: Both methods use squared-error regression on tractable conditional targets to learn marginal quantities.

Connection to diffusion models

1. Recall the forward diffusion process

$$z_t = \sqrt{\alpha_t} x + \sqrt{1 - \alpha_t} \epsilon,$$

2. The conditional distribution is

$$q(z_t \mid x) = \mathcal{N}\left(z_t \mid \sqrt{\alpha_t} x, (1 - \alpha_t) I\right).$$

3. Its score is

$$\boxed{\nabla_{z_t} \ln q(z_t \mid x) = -\frac{\epsilon}{\sqrt{1 - \alpha_t}}.}$$

Connection to diffusion models

1. Suppose a score network predicts $s(z_t, w, t)$.
2. Define a corresponding noise-prediction network by
$$\epsilon_w(z_t, t) = -\sqrt{1-\alpha_t}\, s(z_t, w, t).$$
3. Then
$$\left\| s(z_t, w, t) + \frac{\epsilon}{\sqrt{1-\alpha_t}} \right\|^2 = \frac{1}{1-\alpha_t} \left\| \epsilon_w(z_t, t) - \epsilon \right\|^2.$$
4. Therefore,
$$\boxed{\text{Score prediction} \iff \text{Noise prediction}}$$
up to a noise-level-dependent weighting factor.
5. This explains why the score-matching loss and the noise-prediction loss used in denoising diffusion models have essentially the same structure.
6. The equivalence is exact for a fixed noise level under the corresponding parameterization.
7. When training across different noise levels, the weighting factors must also be accounted for.

Generating samples from a score function

1. How can we generate new samples?
2. The key observation is that the score

$$\mathsf{s}(\mathsf{z}) = \nabla_{\mathsf{z}} \ln p(\mathsf{z})$$

   points toward increasing log probability.
3. One might therefore try gradient ascent:

$$\mathsf{z}_{k+1} = \mathsf{z}_k + \eta \mathsf{s}(\mathsf{z}_k).$$

4. However, this would tend to move samples toward local modes of the distribution rather than generate samples according to the full probability density.
5. To obtain a sampling algorithm, we introduce random noise.

Langevin dynamics

1. A common discretization of overdamped Langevin dynamics is

$$\boxed{z_{k+1} = z_k + \frac{\eta}{2} s(z_k) + \sqrt{\eta}\xi_k,}$$

   where

$$\xi_k \sim \mathcal{N}(0, I).$$

2. The two terms have distinct roles:
   - Score term: Moves samples toward regions of higher probability density.
   - Random term: Introduces stochastic exploration, preventing all samples from simply collapsing to density maxima.
3. The continuous-time version, under this scaling convention, is

$$dZ_t = \frac{1}{2}\nabla_z \ln p(Z_t)\, dt + dW_t,$$

   where W_t is standard Brownian motion.
4. Under suitable regularity and ergodicity conditions, this process has stationary density p.

Why does LD sample the correct distribution?

1. The Fokker-Planck equation associated with this SDE is

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot \left(\frac{1}{2} \rho \nabla \ln p \right) + \frac{1}{2} \Delta \rho,$$

   where $\rho(\mathrm{z}, t)$ is the probability density of the evolving samples.
2. At stationarity, set $\rho = p$. Then

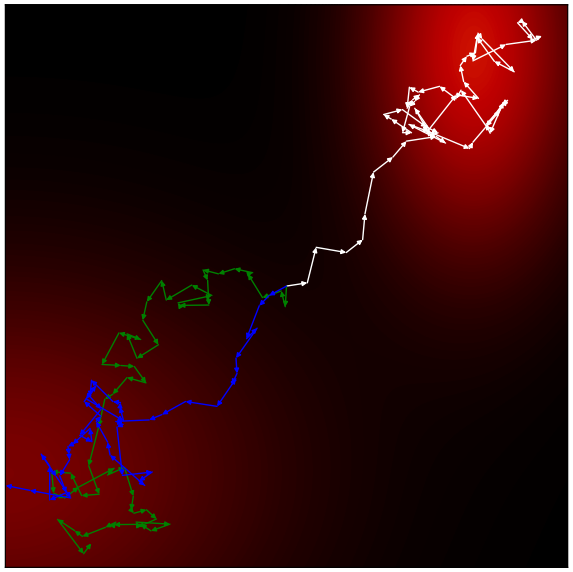
$$\begin{aligned} \frac{\partial p}{\partial t} &= -\frac{1}{2} \nabla \cdot (p \nabla \ln p) + \frac{1}{2} \Delta p \\ &= -\frac{1}{2} \nabla \cdot (\nabla p) + \frac{1}{2} \Delta p \\ &= 0. \end{aligned}$$

3. Thus,

$$\boxed{\rho(\mathrm{z}) = p(\mathrm{z})}$$

   is a stationary solution.
4. The exact continuous-time process therefore preserves the target distribution. A finite-step Euler discretization generally introduces some bias.

Illustration



1. The figure shows several Langevin trajectories, all starting near the center of a two-dimensional distribution.
2. Gradually approach one of the high-density regions. Because of the stochastic term, the paths are irregular.
3. Some trajectories approach the lower-left mode, while another reaches the upper-right mode.
4. Combines directed movement toward high-density regions with random exploration.
5. A difficulty: moving between widely separated modes can be challenging.

Problem 1: Data on a lower-dimensional manifold

1. High-dimensional datasets frequently concentrate near lower-dimensional structures.
2. For example, suppose $\mathrm{x} \in \mathbb{R}^D$ but the data lie approximately on a manifold $\mathcal{M} \subset \mathbb{R}^D$ of intrinsic dimension $d < D$.
3. An idealized data distribution supported exactly on $\mathcal{M}$ may not have a density with respect to D-dimensional Lebesgue measure.
4. Consequently,
$$\nabla_{\mathrm{x}} \ln p(\mathrm{x})$$
may not be well-defined as an ordinary score in the ambient space.
5. Gaussian smoothing addresses this by defining
$$q_\sigma(\mathrm{z}) = \int \mathcal{N}(\mathrm{z} \mid \mathrm{x}, \sigma^2 \mathrm{I}) p(d\mathrm{x}).$$
6. For $\sigma > 0$, this gives a smooth, strictly positive density over $\mathbb{R}^D$, even if the original distribution is singular.

Problem 2: Inaccurate scores in low-density regions

1. The original score-matching loss is

$$J(\mathrm{w}) = \frac{1}{2} \int \|\mathrm{s}(\mathrm{x}, \mathrm{w}) - \nabla_{\mathrm{x}} \ln p(\mathrm{x})\|^2 \, p(\mathrm{x}) \, d\mathrm{x}.$$

2. Notice that the squared error is weighted by $p(\mathrm{x})$.
3. When $p(\mathrm{x})$ is small, errors in the score function contribute relatively little to the loss.
4. Thus, the model may learn the score accurately in high-density regions but poorly in low-density regions.
5. This matters for sampling because Langevin trajectories can pass through low-density regions when moving between modes.
6. An inaccurate score there may cause poor sampling behavior.

Problem 3: Disconnected or widely separated modes

1. Suppose the data distribution is a mixture of two Gaussian components:

$$p(\mathrm{x}) = \frac{1}{2}\mathcal{N}(\mathrm{x} \mid \boldsymbol{\mu}_1, \sigma_0^2 \mathrm{I}) + \frac{1}{2}\mathcal{N}(\mathrm{x} \mid \boldsymbol{\mu}_2, \sigma_0^2 \mathrm{I}).$$

2. If the means are far apart, there may be an extremely low-density region between the components.
3. Langevin sampling can remain trapped near one mode for a long time because transitions between modes require traversing low-probability regions.
4. This is a standard metastability problem in stochastic sampling.

Multiple noise levels

1. Using a sequence of noise variances:

$$\boxed{\sigma_1^2 < \sigma_2^2 < \cdots < \sigma_L^2.}$$

2. Each noise level defines a different smoothed distribution:

$$q_{\sigma_i}(\mathrm{z}) = \int q(\mathrm{z} \mid \mathrm{x}, \sigma_i) p(\mathrm{x})\, d\mathrm{x}.$$

3. At small noise levels, remains close to the original data distribution.
4. At large noise levels, becomes smoother and the gaps between modes become less pronounced.
5. Train a single noise-conditional score network: $\mathrm{s}(\mathrm{z}, \mathrm{w}, \sigma_i^2)$.
6. The objective, averaged over the dataset, is

$$\boxed{J(\mathrm{w}) = \frac{1}{2N} \sum_{n=1}^{N} \sum_{i=1}^{L} \lambda(i) \int \left\| \mathrm{s}(\mathrm{z}, \mathrm{w}, \sigma_i^2) - \nabla_{\mathrm{z}} \ln q(\mathrm{z} \mid \mathrm{x}_n, \sigma_i) \right\|^2 q(\mathrm{z} \mid \mathrm{x}_n, \sigma_i)\, d\mathrm{z}.}$$

Here $\lambda(i)$ is a weighting coefficient for noise level i.

Gaussian corruption

1. For Gaussian corruption,

$$z = x_n + \sigma_i \epsilon,$$

and

$$\nabla_z \ln q(z \mid x_n, \sigma_i) = -\frac{\epsilon}{\sigma_i}.$$

2. Therefore, the objective becomes

$$\boxed{J(w) = \frac{1}{2N} \sum_{n=1}^{N} \sum_{i=1}^{L} \lambda(i) \mathbb{E}_{\epsilon} \left[\left\| s(x_n + \sigma_i \epsilon, w, \sigma_i^2) + \frac{\epsilon}{\sigma_i} \right\|^2 \right].}$$

3. The weighting coefficients are important because the conditional score has magnitude proportional to $1/\sigma_i$.
4. A common choice is

$$\lambda(i) = \sigma_i^2,$$

which helps balance contributions from different noise levels.

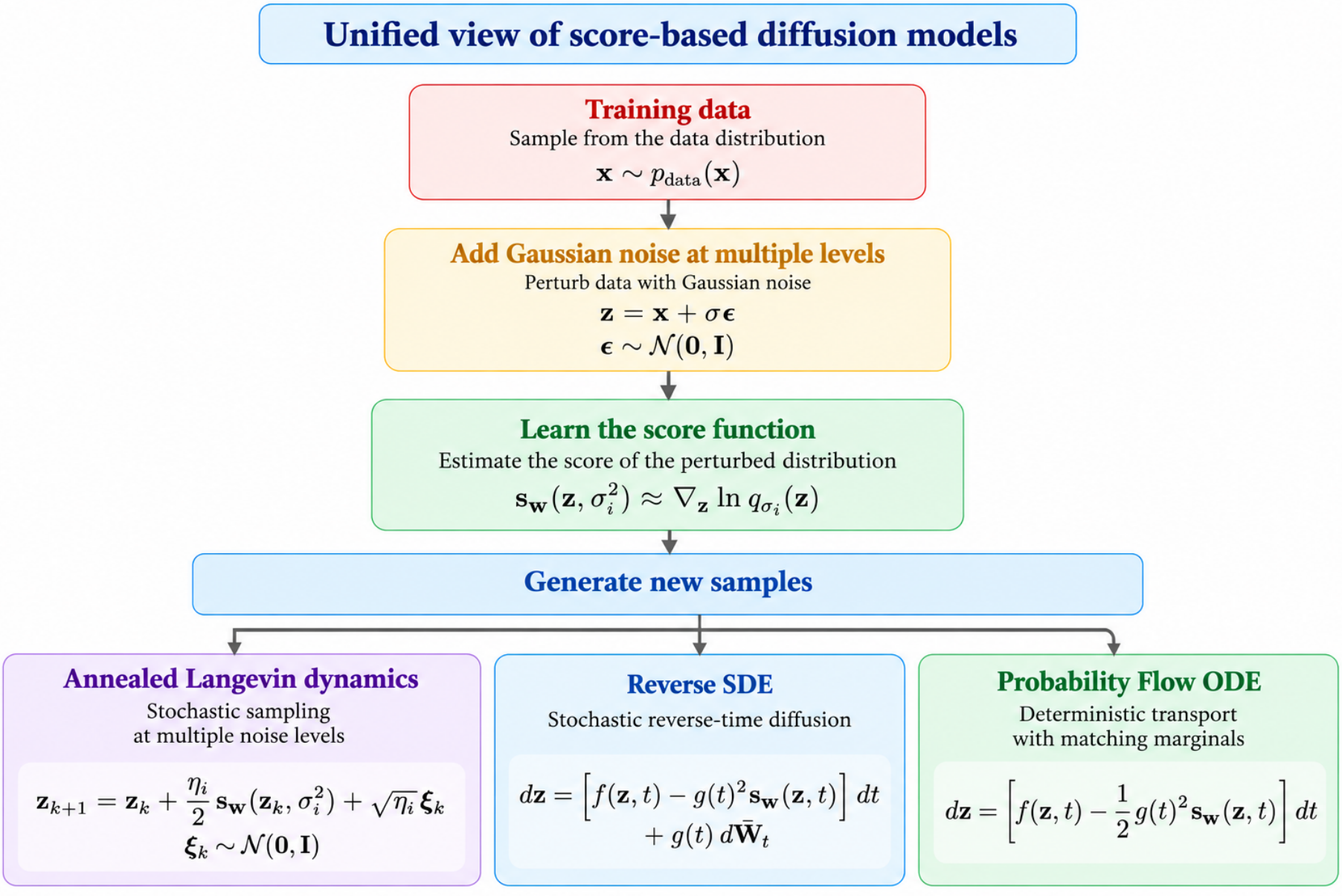
Annealed Langevin dynamics

1. Sampling proceeds from high noise to low noise.
2. For each σ_i, perform several Langevin updates:

$$\boxed{z_{k+1} = z_k + \frac{\eta_i}{2} s(z_k, w, \sigma_i^2) + \sqrt{\eta_i}\xi_k.}$$

3. The procedure is:
   - Initialize z from a broad noise distribution.
   - Start at the largest noise variance σ_L^2.
   - Perform several Langevin sampling using the score at that noise level.
   - Move to the next smaller noise variance.
   - Repeat until reaching the smallest noise variance σ_1^2.
4. The resulting process is called annealed Langevin dynamics.
5. The large-noise stages allow broad exploration, while the small-noise stages refine the samples toward the original data distribution.
6. This is conceptually very close to diffusion-model sampling, where noise is progressively removed.

Putting everything together



From unconditional to conditional diffusion

1. An unconditional diffusion model learns a probability distribution

$$p(\mathrm{x})$$

and generates samples

$$\mathrm{x} \sim p(\mathrm{x}).$$

2. A conditional diffusion model instead aims to sample from

$$\boxed{p(\mathrm{x} \mid \mathrm{c})}$$

where c represents conditioning information.

3. For example: c could be a class label such as "cat." c could be a text prompt such as "a black cat sitting on a chair."
4. A straightforward approach is to train a conditional denoising network

$$\mathrm{g}(z_t, t, \mathrm{c}).$$

5. The denoising network may not sufficiently respect the conditioning information.
6. Guidance addresses this problem by explicitly increasing the influence of c during sampling.

Classifier guidance

1. The score function of a probability density is defined as
$$\boxed{s(x) = \nabla_x \ln p(x).}$$
2. For conditional generation, we want the conditional score
$$\nabla_x \ln p(x \mid c).$$
3. Using Bayes' theorem,
$$p(x \mid c) = \frac{p(c \mid x)p(x)}{p(c)}.$$
$$\ln p(x \mid c) = \ln p(x) + \ln p(c \mid x) - \ln p(c).$$
4. Differentiating with respect to x,
$$\nabla_x \ln p(x \mid c) = \nabla_x \ln p(x) + \nabla_x \ln p(c \mid x),$$
5. Therefore,
$$\boxed{\underbrace{\nabla_x \ln p(x \mid c)}_{\text{Conditional score}} = \underbrace{\nabla_x \ln p(x)}_{\text{Unconditional score}} + \underbrace{\nabla_x \ln p(c \mid x)}_{\text{Classifier gradient}}.}$$

Interpretation

1. The two terms have complementary roles.
2. Unconditional score:
$$\nabla_x \ln p(x)$$
points toward regions where the data distribution has higher probability. It encourages the generation of realistic samples.
3. Classifier gradient:
$$\nabla_x \ln p(c \mid x)$$
points in a direction that increases the classifier's probability of assigning condition c to x.
4. Introducing a guidance scale λ:
$$\boxed{\text{score}(x, c, \lambda) = \nabla_x \ln p(x) + \lambda \nabla_x \ln p(c \mid x).}$$
The parameter λ controls the influence of the conditioning information.

A useful probabilistic interpretation

1. The guided score can also be expressed as

$$\begin{aligned}\text{score}(x, c, \lambda) &= \nabla_x \ln p(x) + \lambda \nabla_x \ln p(c \mid x) \\ &= \nabla_x \ln \left[p(x) p(c \mid x)^\lambda \right].\end{aligned}$$

2. Thus, the guided score corresponds to a modified density

$$\boxed{p_\lambda(x \mid c) \propto p(x) p(c \mid x)^\lambda.}$$

3. When $\lambda = 1$, this becomes the ordinary conditional distribution.
4. When $\lambda > 1$, the classifier probability is raised to a higher power. Samples that strongly satisfy the condition are favored.
5. This explains why stronger guidance can reduce diversity: the modified distribution becomes more concentrated around samples that the classifier recognizes with high confidence.

The classifier must handle noisy inputs

1. The derivation above uses x for simplicity. In an actual diffusion model, guidance is applied at intermediate noise levels.
2. Therefore, the relevant identity is

$$\boxed{\nabla_{z_t} \ln p_t(z_t \mid c) = \nabla_{z_t} \ln p_t(z_t) + \nabla_{z_t} \ln p_t(c \mid z_t).}$$

3. The classifier must estimate

$$p_t(c \mid z_t)$$

   for noisy samples z_t, not just clean images.
4. This is a major limitation of classifier guidance: it requires training a separate noise-aware classifier.

Classifier-free guidance

1. Eliminates the need for a separate classifier.
2. Its key insight is that the classifier gradient can be replaced by the difference between conditional and unconditional scores.
3. From Bayes' theorem,

$$\nabla_x \ln p(c \mid x) = \nabla_x \ln p(x \mid c) - \nabla_x \ln p(x).$$

4. Substituting into the classifier-guidance equation,

$$\text{score}(x, c, \lambda) = \nabla_x \ln p(x) + \lambda \left[\nabla_x \ln p(x \mid c) - \nabla_x \ln p(x)\right].$$

$$\boxed{\text{score}(x, c, \lambda) = \lambda \nabla_x \ln p(x \mid c) + (1 - \lambda) \nabla_x \ln p(x).}$$

5. Define

$$s_{\text{cond}} = \nabla_x \ln p(x \mid c)$$

$$s_{\text{uncond}} = \nabla_x \ln p(x).$$

$$\boxed{s_{\text{CFG}} = s_{\text{uncond}} + \lambda(s_{\text{cond}} - s_{\text{uncond}}).}$$

6. The difference $\boxed{s_{\text{cond}} - s_{\text{uncond}}}$ represents the additional score information introduced by conditioning.

Classifier-free guidance

1. Why can $\lambda > 1$ improve conditioning? Consider

$$\mathsf{s}_{\mathrm{CFG}} = (1-\lambda)\mathsf{s}_{\mathrm{uncond}} + \lambda \mathsf{s}_{\mathrm{cond}}.$$

2. When $0 \leq \lambda \leq 1$, the guided score is a convex combination of the unconditional and conditional scores.
3. When $\lambda > 1$, it is an extrapolation beyond the conditional score.
4. For example, if $\lambda = 3$,

$$\mathsf{s}_{\mathrm{CFG}} = 3\mathsf{s}_{\mathrm{cond}} - 2\mathsf{s}_{\mathrm{uncond}}.$$

5. This does not mean the model simply discards realism. Rather, it emphasizes the difference between conditional and unconditional scores.

How is classifier-free guidance trained?

1. A major advantage of CFG is that one denoising network can learn both conditional and unconditional predictions.
2. Suppose the denoising network is
$$g_\theta(z_t, t, c).$$
3. During training, the conditioning variable is randomly replaced by a null condition $\varnothing$.
4. For example,
$$\widetilde{c} = \begin{cases} c, & \text{with probability } 1 - p_{\text{drop}}, \\ \varnothing, & \text{with probability } p_{\text{drop}}. \end{cases}$$
5. The same network then learns two behaviors:
$$g_\theta(z_t, t, c) \quad \text{(conditional)}$$
and
$$g_\theta(z_t, t, \varnothing) \quad \text{(unconditional)}.$$
6. At inference time, the model is evaluated with both inputs.

How is classifier-free guidance trained?

1. If the network predicts noise, as in DDPM-style diffusion models, the CFG noise prediction is

$$\boxed{\widehat{\boldsymbol{\epsilon}}_{\mathrm{CFG}} = \widehat{\boldsymbol{\epsilon}}_{\mathrm{uncond}} + \lambda\left(\widehat{\boldsymbol{\epsilon}}_{\mathrm{cond}} - \widehat{\boldsymbol{\epsilon}}_{\mathrm{uncond}}\right).}$$

2. This follows from the relationship between the score and the noise prediction.
3. For Gaussian perturbations of the form

$$\mathsf{z}_t = \alpha_t \mathsf{x} + \sigma_t \boldsymbol{\epsilon},$$

   the score estimate is

$$\mathsf{s}_\theta(\mathsf{z}_t, t) = -\frac{\widehat{\boldsymbol{\epsilon}}_\theta(\mathsf{z}_t, t)}{\sigma_t}.$$

4. Because this relationship is linear, the same CFG combination can be applied to noise predictions.

Text-guided image generation

1. The condition c is a text prompt.
2. The text is converted into a representation by a language model or text encoder and supplied to the denoising network.
3. Concatenating text representations with the denoising network's inputs or internal features.
4. Using cross-attention so image features can attend to text tokens. Classifier-free guidance then controls how strongly the generated image follows the prompt.

Image super-resolution

1. The condition is a low-resolution image, and the goal is to generate a corresponding high-resolution image.
2. Mathematically,
$$p(\mathrm{x}_{\mathrm{HR}} \mid \mathrm{x}_{\mathrm{LR}}).$$
3. Cascaded diffusion models that progressively increase image resolution, for example
$$64 \times 64 \longrightarrow 256 \times 256 \longrightarrow 1024 \times 1024.$$
4. Each stage conditions on the output of the previous stage.

Latent diffusion models

1. Rather than performing diffusion directly in high-dimensional image space, latent diffusion models operate in a lower-dimensional representation.
2. An encoder maps an image to a latent representation,

$$\mathsf{h} = E(\mathsf{x}).$$

3. Diffusion is performed in latent space, and a decoder maps the generated latent representation back to image space,

$$\widehat{\mathsf{x}} = D(\widehat{\mathsf{h}}).$$

4. The primary advantage is computational efficiency, because the denoising network operates on lower-dimensional representations.
5. Guidance can be applied in latent space just as it can in image space.

Outline

1 Introduction

2 Flow Matching

3 Diffusion Models

4 Score Matching

5 Connections
- ODE and SDE
- Connections Among Flow, Diffusion, and Score
- Unified Training and Generation

6 Frontiers

7 Latent Variable Models

8 Variational Inference

From deterministic to stochastic trajectories

1. An ODE specifies deterministic evolution once the initial state is fixed:

$$\frac{dX_t}{dt} = b(X_t, t).$$

2. A stochastic differential equation (SDE) adds random perturbations to the dynamics.
3. A stochastic process $(X_t)_{0 \le t \le 1}$ consists of random variables indexed by time:
   - Each X_t is a random variable in $\mathbb{R}^d$.
   - Each realization $t \mapsto X_t$ is a sample trajectory.
4. Repeating a stochastic simulation can yield different trajectories, even from the same initial state.

Brownian motion

1. Brownian motion, or a Wiener process, $(W_t)_{t\geq 0}$ is a continuous stochastic process with $W_0 = 0$.
2. It can be viewed as a continuous random walk and originated in the study of physical diffusion.
3. Its increments are Gaussian:

$$W_t - W_s \sim \mathcal{N}(0, (t-s)I_d), \qquad 0 \leq s < t.$$

4. Increments over disjoint time intervals are independent.
5. Its increments are stationary: their distributions depend only on the elapsed time.
6. Brownian motion is as fundamental to stochastic processes as the Gaussian distribution is to probability distributions.
7. Brownian paths are continuous but almost surely nowhere differentiable; in particular, they have infinite variation and infinite length on nontrivial intervals.

Brownian increments and simulation

1. For d-dimensional standard Brownian motion,

$$\mathbb{E}[W_t] = 0, \quad \mathrm{Cov}(W_t) = tI_d, \quad \mathrm{Cov}(W_t, W_s) = \min(t, s) I_d.$$

2. Brownian motion itself is not stationary, but its increments are stationary.
3. For a time step $h > 0$,

$$W_{t+h} - W_t \overset{d}{=} \sqrt{h}\,\epsilon_t, \qquad \epsilon_t \sim \mathcal{N}(0, I_d).$$

4. Simulate on a time grid using $W_0 = 0$ and

$$W_{t+h} = W_t + \sqrt{h}\,\epsilon_t,$$

with independent noises at successive steps.

From Brownian motion to SDEs

1. An Itô SDE with time-dependent scalar diffusion is

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

2. The drift b describes systematic motion; $g(t)$ controls random fluctuations.
3. Over a short interval h,

$$X_{t+h} - X_t \approx b(X_t, t)h + g(t)\sqrt{h}\,\epsilon, \qquad \epsilon \sim \mathcal{N}(0, I_d).$$

4. The conditional local moments satisfy

$$\mathbb{E}[\Delta X_t \mid X_t = x] = b(x, t)h + o(h),$$
$$\mathrm{Cov}(\Delta X_t \mid X_t = x) = g^2(t)hI_d + o(h).$$

Why Brownian noise scales as $\sqrt{dt}$

1. The notation $dW_t \sim \mathcal{N}(0, dt\, I_d)$ is shorthand motivated by finite increments:
$$W_{t+h} - W_t \sim \mathcal{N}(0, hI_d).$$
2. Equivalently, $\Delta W_t \overset{d}{=} \sqrt{h}\,\epsilon$, with $\epsilon \sim \mathcal{N}(0, I_d)$.
3. The two terms in an SDE scale differently:
$$\underbrace{b(X_t, t)h}_{\text{order } h} + \underbrace{g(t)\sqrt{h}\,\epsilon}_{\text{order } \sqrt{h}}.$$
4. Since Brownian paths are almost surely nowhere differentiable, dW_t/dt is not an ordinary derivative; SDEs are interpreted through stochastic integrals.

Euler–Maruyama discretization

1. Consider $dX_t = b(X_t, t)\,dt + g(t)\,dW_t$.
2. Approximate coefficients by their values at the beginning of a time step h:
$$X_{t+h} \approx X_t + b(X_t, t)h + g(t)(W_{t+h} - W_t).$$
3. Using Gaussian Brownian increments gives
$$\boxed{X_{k+1} = X_k + b(X_k, t_k)h + g(t_k)\sqrt{h}\,\epsilon_k,}$$
where $\epsilon_k \sim \mathcal{N}(0, I_d)$ are independent.
4. For comparison, Euler's method for an ODE omits the random term:
$$X_{k+1} = X_k + b(X_k, t_k)h.$$
5. Repeated simulations approximate the evolving distribution of the SDE.

Continuity equation for deterministic flows

1. Let $dX_t/dt = v_t(X_t)$ and $X_0 \sim p_0$.
2. If $p_t(x)$ is the density of X_t, then

$$\boxed{\frac{\partial p_t}{\partial t} = -\nabla \cdot (p_t v_t).}$$

3. The equation expresses conservation of probability:
   - $\partial_t p_t$ measures the local density change.
   - $p_t v_t$ is the probability flux.
   - $-\nabla \cdot (p_t v_t)$ measures net probability inflow.
4. Probability mass is transported by the velocity field without being created or destroyed.

From stochastic trajectories to probability densities

1. Consider the Itô SDE

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

2. The drift b describes systematic motion; Brownian increments introduce random fluctuations in each sample trajectory.
3. The random state X_t has a time-dependent density $p_t(x)$, assuming such a density exists.
4. The SDE describes individual random trajectories, while the Fokker–Planck equation describes the deterministic evolution of their probability density.
5. Our next question is how b and g determine the change of p_t.

The Fokker–Planck equation

1. For $dX_t = b(X_t, t)\,dt + g(t)\,dW_t$ with $X_t \sim p_t$,

$$\boxed{\partial_t p_t = -\nabla \cdot (b p_t) + \frac{g^2(t)}{2} \Delta p_t.}$$

2. The drift term $-\nabla \cdot (b p_t)$ transports probability with the deterministic field b.
3. The diffusion term $\frac{g^2(t)}{2} \Delta p_t$ redistributes probability through Brownian fluctuations; $\Delta = \nabla \cdot \nabla$ is the Laplacian.
4. When $g(t) = 0$, the equation reduces to the continuity equation for deterministic flows.
5. When $b = 0$ and g is constant, it becomes the heat equation, describing diffusive spreading.

Understanding probability flux

1. The Fokker–Planck equation can be written as probability conservation:

$$\partial_t p_t = -\nabla \cdot J_t, \qquad J_t = \underbrace{b p_t}_{\text{drift flux}} - \underbrace{\frac{g^2(t)}{2} \nabla p_t}_{\text{diffusive flux magnitude}}.$$

2. Drift flux carries probability in the direction of b; diffusive flux $-\frac{g^2(t)}{2} \nabla p_t$ points down density gradients.
3. For a region Ω with outward unit normal n,

$$\frac{d}{dt} \int_\Omega p_t(x)\, dx = -\int_{\partial\Omega} J_t \cdot n\, dS.$$

4. Thus, probability in Ω changes through its boundary; suitable boundary conditions conserve total probability.

Deriving Fokker–Planck using Itô's formula

1. Let $\varphi(x)$ be a smooth, compactly supported test function. Itô's formula gives

$$d\varphi(X_t) = \nabla\varphi(X_t)^\top b(X_t, t)\, dt + \frac{g^2(t)}{2}\Delta\varphi(X_t)\, dt + g(t)\nabla\varphi(X_t)^\top dW_t.$$

2. Taking expectations, the Itô stochastic integral has zero mean (under standard integrability assumptions):

$$\frac{d}{dt}\mathbb{E}[\varphi(X_t)] = \mathbb{E}\left[b(X_t, t)^\top \nabla\varphi(X_t) + \frac{g^2(t)}{2}\Delta\varphi(X_t)\right].$$

3. This expresses the evolution of all smooth observables in terms of the SDE coefficients.

From the weak form to Fokker–Planck

1. Express expectations through the density p_t:

$$\int \varphi(x)\, \partial_t p_t(x)\, dx = \int \left[b^\top \nabla \varphi + \frac{g^2(t)}{2} \Delta \varphi \right] p_t\, dx.$$

2. Integrating by parts transfers derivatives from φ to p_t:

$$\int \varphi\, \partial_t p_t\, dx = \int \varphi \left[-\nabla \cdot (b p_t) + \frac{g^2(t)}{2} \Delta p_t \right] dx.$$

3. Because this holds for every smooth test function,

$$\boxed{\partial_t p_t = -\nabla \cdot (b p_t) + \frac{g^2(t)}{2} \Delta p_t.}$$

4. The derivation assumes sufficient regularity and vanishing boundary terms (or an appropriate weak interpretation).

Fokker–Planck as a continuity equation

1. Recall the probability flux $J_t = bp_t - \frac{g^2(t)}{2}\nabla p_t$.
2. Wherever $p_t(x) > 0$, the identity $\nabla p_t = p_t \nabla \log p_t$ yields

$$\begin{aligned}\partial_t p_t &= -\nabla \cdot (bp_t) + \frac{g^2(t)}{2}\Delta p_t \\ &= -\nabla \cdot \left[p_t \left(b - \frac{g^2(t)}{2}\nabla \log p_t \right) \right].\end{aligned}$$

3. This has the continuity-equation form $\partial_t p_t = -\nabla \cdot (p_t v_t)$, with effective velocity

$$\boxed{v_t(x) = b(x,t) - \frac{g^2(t)}{2}\nabla \log p_t(x).}$$

4. The score correction accounts for the probability spreading induced by Brownian noise.

Connecting ODEs and SDEs

1. Suppose an ODE $dX_t/dt = v_t(X_t)$ generates a known probability path p_t.
2. For any prescribed scalar diffusion coefficient $g(t)$, consider

$$\boxed{dY_t = \left[v_t(Y_t) + \frac{g^2(t)}{2}\nabla \log p_t(Y_t)\right] dt + g(t)\, dW_t.}$$

3. The extra score drift compensates for the spreading induced by the added Brownian noise.
4. The resulting SDE and original ODE share the same marginal probability path, but generally have different trajectories.
5. Setting $g(t) = 0$ recovers the original ODE.

Why do diffusion models need reverse-time dynamics?

1. Forward diffusion gradually maps a complex data distribution to simple noise:
$$X_0 \sim p_{\text{data}} \quad \longrightarrow \quad X_T \approx \mathcal{N}(0, I_d).$$
2. Adding Gaussian noise is straightforward, but generation requires the opposite evolution:
$$X_T \sim \mathcal{N}(0, I_d) \quad \longrightarrow \quad X_0 \sim p_{\text{data}}.$$
3. A reverse-time SDE evolves the marginal distributions backward; it does not simply retrace individual forward trajectories.
4. Brownian noise remains present in reverse time, so reversing the deterministic drift alone cannot generally reverse the probability evolution.

Reverse-time SDE and the score

1. Start with the forward SDE

$$dX_t = b(X_t, t)\,dt + g(t)\,dW_t, \qquad X_t \sim p_t.$$

2. For decreasing physical time t (so $dt < 0$), the reverse-time SDE is

$$\boxed{dX_t = \left[b(X_t, t) - g^2(t)\nabla \log p_t(X_t)\right] dt + g(t)\,d\overline{W}_t.}$$

3. The score $\nabla \log p_t(x) = \nabla p_t(x)/p_t(x)$ describes the local variation of the log density and points toward increasing density.
4. Because $dt < 0$, the score correction in the reverse-time drift helps guide samples toward higher-density regions.
5. The score-dependent drift corrects for the spreading caused by Brownian noise; a learned score makes reverse-time sampling possible.
6. $\overline{W}_t$ denotes Brownian noise associated with backward-time integration.

Deriving the reverse-time drift

1. Introduce increasing reverse time $s = T - t$, with $Y_s = X_{T-s}$ and $q_s = p_{T-s}$.
2. The forward Fokker–Planck equation implies

$$\frac{\partial q_s}{\partial s} = \nabla \cdot (q_s b(\cdot, T - s)) - \frac{g^2(T - s)}{2} \Delta q_s.$$

3. Suppose $dY_s = \widetilde{b}(Y_s, s)\, ds + g(T - s)\, d\overline{W}_s$. Its density obeys

$$\frac{\partial q_s}{\partial s} = -\nabla \cdot (q_s \widetilde{b}) + \frac{g^2(T - s)}{2} \Delta q_s.$$

4. Comparing both expressions, one valid reverse drift is

$$\boxed{\widetilde{b}(x, s) = -b(x, T - s) + g^2(T - s) \nabla \log q_s(x).}$$

Reverse-time Brownian motion

1. With reverse time $s = T - t$, the reverse process is written with s increasing from 0 to T.
2. The driving noise $\overline{W}_s$ is standard Brownian motion in this reverse-time parameter:
$$\overline{W}_{s+h} - \overline{W}_s \sim \mathcal{N}(0, hI_d), \qquad h > 0.$$
3. Its increments over disjoint reverse-time intervals are independent.
4. Reversing an SDE changes the drift but not the Gaussian character or $\sqrt{h}$ scaling of its stochastic increments.
5. In general, the reverse-time driving Brownian motion should not be identified with simply negating the original Brownian path.

Example: reversing pure Brownian motion

1. Consider $dX_t = \sigma\, dW_t$, $X_0 = 0$, for $0 < t \le T$.
2. Its density and score are

$$p_t = \mathcal{N}(0, \sigma^2 t I_d), \qquad \nabla \log p_t(x) = -\frac{x}{\sigma^2 t}.$$

3. The reverse process $Y_s = X_{T-s}$ satisfies

$$\boxed{dY_s = -\frac{Y_s}{T-s}\, ds + \sigma\, d\overline{W}_s, \quad 0 \le s < T.}$$

4. Simply reversing the zero drift would leave pure Brownian motion and increasing variance.
5. The score-induced restoring drift instead shrinks the variance toward zero, recovering the point mass at the origin as $s \uparrow T$.

Why the reverse SDE has twice the score coefficient

1. For the same forward SDE, compare two dynamics:

$$\text{Probability flow ODE:} \quad \frac{dX_t}{dt} = b - \frac{g^2(t)}{2}\nabla \log p_t,$$

$$\text{Reverse-time SDE:} \quad dX_t = [b - g^2(t)\nabla \log p_t]dt + g(t)d\bar{W}_t,$$

   where the reverse SDE uses $dt < 0$.

2. The probability flow ODE has no stochastic noise. Its score term accounts for the entire diffusion effect through an effective deterministic velocity.
3. The reverse SDE still contains Brownian noise. Its drift must compensate for this noise while reversing the probability evolution, requiring the full score coefficient.
4. Both dynamics reproduce the same marginal distributions p_t when the exact score is used. They differ in their individual trajectories, not their marginal density evolution.

From discrete diffusion to continuous-time SDEs

1. A discrete Gaussian forward diffusion step has the form

$$z_{k+1} = \sqrt{1-\beta_k}\, z_k + \sqrt{\beta_k}\, \epsilon_k, \qquad \epsilon_k \sim \mathcal{N}(0, I_d).$$

2. With a small time step h and $\beta_k \approx \beta(t_k)h$,

$$\sqrt{1-\beta_k} \approx 1 - \frac{1}{2}\beta(t_k)h.$$

3. Consequently,

$$z_{k+1} - z_k \approx -\frac{1}{2}\beta(t_k)z_k h + \sqrt{\beta(t_k)}\sqrt{h}\,\epsilon_k.$$

4. This is Euler–Maruyama for the variance-preserving SDE

$$\boxed{dZ_t = -\frac{1}{2}\beta(t)Z_t\, dt + \sqrt{\beta(t)}\, dW_t.}$$

Continuous-time variance-preserving diffusion

1. The discrete forward process uses a Gaussian transition:

$$z_{k+1} = \sqrt{1-\beta_k}\, z_k + \sqrt{\beta_k}\, \epsilon_k.$$

2. Under $\beta_k \approx \beta(t_k)h$, the limiting SDE has drift and diffusion

$$f(z,t) = -\frac{1}{2}\beta(t)z, \qquad g(t) = \sqrt{\beta(t)}.$$

3. Its conditional distribution retains the same Gaussian corruption structure:

$$Z_t \mid Z_0 = z_0 \sim \mathcal{N}\left(e^{-\frac{1}{2}\int_0^t \beta(s)\,ds} z_0, \left(1 - e^{-\int_0^t \beta(s)\,ds}\right) I_d\right).$$

4. Hence continuous-time diffusion and discrete-time diffusion describe closely related Gaussian noise schedules, not different state spaces.

Variance-preserving SDE and Gaussian corruption

1. The variance-preserving (VP) SDE is

$$dZ_t = -\frac{1}{2}\beta(t)Z_t\,dt + \sqrt{\beta(t)}\,dW_t.$$

2. Let $B(t) = \int_0^t \beta(s)\,ds$. Its conditional distribution is

$$\boxed{p(z_t \mid z_0) = \mathcal{N}\left(e^{-B(t)/2}z_0, \left(1 - e^{-B(t)}\right) I_d\right).}$$

3. The discrete diffusion counterpart is

$$q(z_k \mid z_0) = \mathcal{N}\left(\sqrt{\bar{\alpha}_k}\, z_0, (1 - \bar{\alpha}_k) I_d\right),$$

   where $\bar{\alpha}_k = \prod_{j=1}^{k}(1 - \beta_j)$.

4. Thus $e^{-B(t)}$ plays the continuous-time role of $\bar{\alpha}_k$.

Reverse variance-preserving SDE

1. For the VP SDE,

$$f(z,t) = -\frac{1}{2}\beta(t)z, \qquad g^2(t) = \beta(t).$$

2. Its reverse-time dynamics, integrated with $dt < 0$, are

$$\boxed{dZ_t = \left[-\frac{1}{2}\beta(t)Z_t - \beta(t)\nabla_z \log p_t(Z_t)\right] dt + \sqrt{\beta(t)}\, d\overline{W}_t.}$$

3. Replace the unknown score with a neural approximation $s_\theta(z,t)$:

$$dZ_t = \left[-\frac{1}{2}\beta(t)Z_t - \beta(t)s_\theta(Z_t,t)\right] dt + \sqrt{\beta(t)}\, d\overline{W}_t.$$

4. Starting at the noise endpoint and integrating backward generates approximate data samples.

Probability flow ODE for VP diffusion

1. The VP SDE has a deterministic probability flow ODE:

$$\boxed{\frac{dZ_t}{dt} = -\frac{1}{2}\beta(t)Z_t - \frac{1}{2}\beta(t)\nabla_z \log p_t(Z_t).}$$

2. With a learned score, replace $\nabla_z \log p_t$ by s_θ.
3. Compare the score terms:
   - Reverse VP SDE: $-\beta(t)\nabla_z \log p_t$ in its drift.
   - VP probability flow ODE: $-\frac{1}{2}\beta(t)\nabla_z \log p_t$.
4. Both dynamics describe the same marginal probability path under exact modeling, but differ in stochasticity and sample trajectories.

Connection to normalizing flows

1. The probability flow ODE is a continuous normalizing flow:

$$\frac{dz_t}{dt} = v(z_t, t), \qquad \partial_t p_t = -\nabla \cdot (p_t v).$$

2. Expand the divergence in the continuity equation:

$$\partial_t p_t = -v^\top \nabla p_t - p_t \nabla \cdot v,$$
$$\partial_t \log p_t = -v^\top \nabla \log p_t - \nabla \cdot v.$$

3. Apply the chain rule along the ODE trajectory:

$$\frac{d}{dt} \log p_t(z_t) = \partial_t \log p_t(z_t) + \left(\frac{dz_t}{dt}\right)^\top \nabla \log p_t(z_t)$$
$$= -\nabla \cdot v(z_t, t).$$

4. This is the instantaneous change-of-variables formula.

Likelihood evaluation along the probability flow ODE

1. Integrate the instantaneous change-of-variables formula from data time 0 to noise time T:

$$\log p_T(z_T) - \log p_0(z_0) = -\int_0^T \nabla \cdot v(z_t, t)\, dt.$$

2. Rearranging gives the data log likelihood:

$$\boxed{\log p_0(z_0) = \log p_T(z_T) + \int_0^T \nabla \cdot v(z_t, t)\, dt.}$$

3. p_T is a tractable terminal noise density, and z_t follows the probability flow ODE from z_0.
4. With an exact score and exact ODE solution, this expression is exact for the corresponding probability path.
5. With learned scores and numerical integration, the computed likelihood approximates the learned model's likelihood; it need not equal the true data likelihood.

Probability flow ODE and its implications

1. The corresponding deterministic equation is called the probability flow ODE:

$$\boxed{\frac{dX_t}{dt} = f_t(X_t) - \frac{g_t^2}{2}\nabla \log p_t(X_t).}$$

2. The SDE and probability flow ODE generally have different trajectories, but can share the same marginal distributions.

Diffusion SDE	Probability Flow ODE
Stochastic trajectories	Deterministic trajectories
Driven by Brownian motion	Driven by a velocity field
Fokker–Planck equation	Continuity equation
Produces $p_t(x)$	Produces the same $p_t(x)$

3. Connection to generative modeling:
   - Score-based diffusion models learn the score to construct stochastic or deterministic dynamics.
   - Flow matching directly learns a velocity field for deterministic generation.
   - Both can describe the same probability evolution, although trajectories and training objectives may differ.

The central mathematical relationship

$$\begin{aligned}
\text{Forward SDE:} \quad & dZ_t = \mathrm{f}\, dt + g\, dW_t, \\
\text{Reverse SDE:} \quad & dZ_t = (\mathrm{f} - g^2 \nabla \ln p_t)\, dt + g\, d\overline{W}_t, \\
\text{Probability flow ODE:} \quad & \frac{dz}{dt} = \mathrm{f} - \frac{1}{2} g^2 \nabla \ln p_t.
\end{aligned}$$

These three equations unify much of the mathematical theory behind modern score-based diffusion models.

Gaussian probability paths: conditional and marginal

1. Use the noise-to-data convention: $t = 0$ is noise and $t = 1$ is data. Let $X_1 \sim p_{\text{data}}$ and $\epsilon \sim \mathcal{N}(0, I)$ be independent.
2. A Gaussian interpolation is

$$X_t = \alpha_t X_1 + \sigma_t \epsilon, \qquad p_t(x \mid x_1) = \mathcal{N}(x; \alpha_t x_1, \sigma_t^2 I).$$

3. The schedules α_t, σ_t determine the amount of signal and noise. To connect standard Gaussian noise to data, choose

$$\alpha_0 = 0, \quad \sigma_0 = 1, \qquad \alpha_1 = 1, \quad \sigma_1 = 0.$$

4. Averaging over data samples yields the marginal path

$$p_t(x) = \int p_t(x \mid x_1) p_{\text{data}}(x_1)\, dx_1.$$

The conditional distribution is Gaussian, but the marginal is generally not.

Linear interpolation as a Gaussian probability path

1. A simple schedule is $\alpha_t = t$ and $\sigma_t = 1 - t$, giving

$$X_t = (1 - t)X_0 + tX_1, \qquad X_0 \sim \mathcal{N}(0, I).$$

2. For a fixed data point x_1, the conditional path is

$$p_t(x \mid x_1) = \mathcal{N}\big(x; tx_1, (1 - t)^2 I\big).$$

3. Each sampled pair (X_0, X_1) defines a straight-line interpolation with instantaneous velocity

$$\dot{X}_t = X_1 - X_0.$$

4. This is a common flow-matching construction, but not the only one: other Gaussian schedules also define valid conditional paths.
5. A probability path specifies the distribution at each time, not a unique motion for individual samples.

Flow matching learns a velocity for the probability path

1. For $X_t = \alpha_t X_1 + \sigma_t \epsilon$, the sampled-path velocity is

$$\dot{X}_t = \dot{\alpha}_t X_1 + \dot{\sigma}_t \epsilon.$$

2. Flow matching trains a velocity model by regression:

$$\mathcal{L}_{\mathrm{FM}}(\theta) = \mathbb{E}_{t, X_1, \epsilon}\left[\left\| v_\theta(X_t, t) - (\dot{\alpha}_t X_1 + \dot{\sigma}_t \epsilon)\right\|^2\right].$$

3. The optimal predictor is the conditional expectation

$$v_t^*(x) = \mathbb{E}[\dot{X}_t \mid X_t = x].$$

4. This velocity transports the marginal densities through

$$\frac{dX_t}{dt} = v_t^*(X_t), \qquad \partial_t p_t = -\nabla \cdot (p_t v_t^*).$$

5. Training uses sampled conditional paths; generation integrates the learned marginal velocity field.

The score and velocity are linked by Gaussian paths

1. For $\sigma_t > 0$, the conditional Gaussian score is

$$\nabla_x \log p_t(x \mid x_1) = -\frac{x - \alpha_t x_1}{\sigma_t^2}.$$

2. Averaging over the posterior distribution of X_1 gives the marginal score:

$$\nabla_x \log p_t(x) = \mathbb{E}[\nabla_x \log p_t(x \mid X_1) \mid X_t = x] = -\frac{1}{\sigma_t}\mathbb{E}[\epsilon \mid X_t = x].$$

3. The flow-matching velocity is

$$v_t^*(x) = \dot{\alpha}_t \mathbb{E}[X_1 \mid X_t = x] + \dot{\sigma}_t \mathbb{E}[\epsilon \mid X_t = x].$$

4. Thus both the velocity and the score are determined by the same Gaussian conditional path and posterior averages.
5. They are different prediction targets, but are mathematically related; for $\alpha_t > 0$ the velocity can be expressed using the score.

An explicit velocity–score relationship

1. Since $X_t = \alpha_t X_1 + \sigma_t \epsilon$, conditioning on $X_t = x$ yields

$$\mathbb{E}[X_1 \mid X_t = x] = \frac{x - \sigma_t \mathbb{E}[\epsilon \mid X_t = x]}{\alpha_t}.$$

2. Substitute this and $\mathbb{E}[\epsilon \mid X_t = x] = -\sigma_t \nabla \log p_t(x)$ into the velocity:

$$\boxed{v_t^*(x) = \frac{\dot{\alpha}_t}{\alpha_t} x - \sigma_t \left(\dot{\sigma}_t - \frac{\dot{\alpha}_t}{\alpha_t} \sigma_t \right) \nabla \log p_t(x).}$$

3. For linear interpolation $\alpha_t = t$, $\sigma_t = 1 - t$ (with $0 < t < 1$),

$$\boxed{v_t^*(x) = \frac{x}{t} + \frac{1 - t}{t} \nabla \log p_t(x).}$$

4. The formulas show how a score field determines a velocity field for the same Gaussian path, away from singular endpoints.

Flow matching versus score-based diffusion

1. Both methods can describe a path between a simple noise distribution and the target data distribution, but use different learning targets:
   - Flow matching learns a velocity $v_t(x)$, typically through conditional velocity regression.
   - Score-based diffusion learns $\nabla \log p_t(x)$ (or an equivalent noise/denoising prediction).
2. Their generative dynamics differ:
   - Flow matching typically samples using a deterministic ODE.
   - Score-based diffusion samples using a reverse SDE or a probability flow ODE.
3. When both are constructed for the same marginal path, a suitable velocity field and score field are related, and the corresponding ODEs can reproduce the same marginals.
4. The marginal path alone does not uniquely determine the stochastic process or the individual sample trajectories.
5. The connection is at the level of probability evolution; it does not imply identical training objectives, schedules, or sample trajectories in every implementation.

Flow matching versus diffusion models

Aspect	Flow Matching	Score-Based Diffusion
Learned quantity	Velocity field $v_\theta(x, t)$	Score $\nabla_x \log p_t(x)$ or equivalent denoising target
Generative dynamics	Usually deterministic ODE	Reverse-time SDE or probability-flow ODE
Brownian noise during sampling	Not required	Required for SDE sampling (not ODE)
Training target	Conditional velocity	Conditional score or noise
Probability path	Flexible	Typically derived from a noising SDE
Training	Often simulation-free regression	Often denoising regression
Sampling	Numerical ODE integration	Numerical SDE or ODE integration

1. Flow matching and diffusion models are closely related through their probability paths.
2. They can describe the same evolution of probability distributions while using different formulations for learning and generating sample trajectories.

Flow matching and score matching

1. Score matching learns the gradient of the log probability density, which describes the local geometry of a probability distribution.
2. Flow matching learns a velocity field, which describes how probability mass moves as the distribution evolves over time.
3. For Gaussian probability paths, we can derive an explicit mathematical relationship between the score and the velocity field.

Flow matching and score matching

1. Both methods describe probability distributions, but from different perspectives.
   - Score matching: Learns how log probability density changes with position.
   - Flow matching: Learns how samples move as the probability distribution evolves over time.
2. Their central quantities are:
   - Score Function
   $$s_t(x) = \nabla_x \log p_t(x).$$
   - Velocity Field
   $$\frac{dx_t}{dt} = v_t(x_t).$$
3. Key distinction: The score describes the spatial structure of a distribution at a given time, while velocity describes how probability mass moves over time.

Gaussian probability path and conditional score

1. Consider a Gaussian probability path:

$$\boxed{X_t = \alpha_t X_1 + \sigma_t \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I).}$$

Here, X_1 is sampled from the data distribution, and α_t, σ_t control the contributions of data and noise.

2. Conditioning on $X_1 = x_1$ gives

$$p_t(x \mid x_1) = \mathcal{N}(x; \alpha_t x_1, \sigma_t^2 I).$$

The conditional score is

$$\boxed{\nabla_x \ln p_t(x \mid x_1) = -\frac{x - \alpha_t x_1}{\sigma_t^2}.}$$

3. Score matching learns the marginal score by averaging conditional scores:

$$\boxed{s_t(x) = \mathbb{E}_{X_1 \mid X_t = x} \left[\nabla_x \ln p_t(x \mid X_1) \right].}$$

This follows from the conditional expectation property of squared-error regression.

Gaussian probability path and conditional velocity

1. Flow matching uses the same Gaussian probability path:

$$X_t = \alpha_t X_1 + \sigma_t \epsilon.$$

Keeping X_1 and ϵ fixed, differentiating with respect to time gives

$$\boxed{\dot{X}_t = \dot{\alpha}_t X_1 + \dot{\sigma}_t \epsilon.}$$

2. Since $\epsilon = \frac{x - \alpha_t x_1}{\sigma_t}$, the conditional velocity field is

$$\boxed{u_t(x \mid x_1) = \dot{\alpha}_t x_1 + \frac{\dot{\sigma}_t}{\sigma_t}(x - \alpha_t x_1).}$$

The two terms describe changes in the data contribution and noise scale, respectively.

3. Flow matching learns the marginal velocity:

$$\boxed{v_t(x) = \mathbb{E}_{X_1 \mid X_t = x}[u_t(x \mid X_1)].}$$

This uses the same conditional expectation principle as score matching, but with velocity targets.

Deriving the relationship between score and velocity

1. Recall the conditional score:

$$s_t(x \mid x_1) = -\frac{x - \alpha_t x_1}{\sigma_t^2}.$$

Rearranging gives

$$\boxed{\alpha_t x_1 = x + \sigma_t^2 s_t(x \mid x_1).}$$

2. Substitute into the conditional velocity:

$$\begin{aligned} u_t(x \mid x_1) &= \dot{\alpha}_t x_1 + \frac{\dot{\sigma}_t}{\sigma_t}(x - \alpha_t x_1) \\ &= \frac{\dot{\alpha}_t}{\alpha_t}\left(x + \sigma_t^2 s_t(x \mid x_1)\right) - \sigma_t \dot{\sigma}_t s_t(x \mid x_1) \\ &= \frac{\dot{\alpha}_t}{\alpha_t} x + \left(\frac{\dot{\alpha}_t}{\alpha_t}\sigma_t^2 - \sigma_t \dot{\sigma}_t\right) s_t(x \mid x_1). \end{aligned}$$

3. Define the time-dependent coefficients: $a_t = \frac{\dot{\alpha}_t}{\alpha_t}, \qquad b_t = a_t \sigma_t^2 - \sigma_t \dot{\sigma}_t.$
Therefore,

$$\boxed{u_t(x \mid x_1) = a_t x + b_t s_t(x \mid x_1).}$$

From conditional fields to marginal fields

1. Recall the relationship between conditional velocity and score:

$$u_t(x \mid x_1) = a_t x + b_t s_t(x \mid x_1).$$

2. Take the conditional expectation given $X_t = x$:

$$\begin{aligned} v_t(x) &= \mathbb{E}[u_t(x \mid X_1) \mid X_t = x] \\ &= \mathbb{E}[a_t x + b_t s_t(x \mid X_1) \mid X_t = x] \\ &= a_t x + b_t \mathbb{E}[s_t(x \mid X_1) \mid X_t = x]. \end{aligned}$$

3. Since the posterior average of conditional scores equals the marginal score,

$$\boxed{v_t(x) = a_t x + b_t s_t(x).}$$

4. When $b_t \neq 0$, the relationship can be inverted:

$$\boxed{s_t(x) = \frac{v_t(x) - a_t x}{b_t}.}$$

Key insight: For a specified Gaussian probability path, score and velocity are related by a time-dependent affine transformation.

Linear interpolation: a simple example

1. Consider linear-interpolation flow matching:

$$\boxed{X_t = (1-t)X_0 + tX_1, \qquad X_0 \sim \mathcal{N}(0, I).}$$

   This corresponds to $\alpha_t = t, \qquad \sigma_t = 1 - t.$

2. The score-velocity relationship simplifies to:

$$\boxed{v_t(x) = \frac{x + (1-t)s_t(x)}{t}, \qquad 0 < t < 1.}$$

   Thus, the velocity can be computed from the score, and vice versa.

3. Broader implications:

Prediction Approach	**Learning Target**
Score prediction	Gradient of log density
Noise prediction	Gaussian noise estimate
Data prediction	Clean data estimate
Velocity prediction	Sample transport velocity

4. These prediction targets are mathematically related under Gaussian probability paths, although their training objectives may differ.
5. Main Takeaway: Gaussian probability paths connect flow matching, score matching, denoising, and diffusion models.

Comparison of training objectives

1. Flow matching, diffusion models, and score matching can all be trained using supervised regression with analytically available targets.
2. Their main differences are summarized below:

	Flow matching	Diffusion models	Score matching
Network predicts	Velocity	Noise	Score
Training target	Conditional velocity	Gaussian noise	Conditional score
Loss function	MSE	MSE	MSE
Training input	Sample, time	Sample, time	Sample, time/noise level
Generation	ODE	Reverse denoising	Reverse SDE / ODE

3. The three methods differ primarily in *what the network predicts*, rather than the basic training procedure.

Unified training framework

1. All three methods can use a Gaussian probability path:

$$\boxed{\mathsf{x}_t = \alpha_t \mathsf{x}_1 + \sigma_t \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathsf{I}).}$$

2. Their training objectives share the same form:

$$\boxed{\mathcal{L}(\theta) = \mathbb{E}_{t, \mathsf{x}_1, \boldsymbol{\epsilon}} \left[w(t) \left\| \mathsf{f}_\theta(\mathsf{x}_t, t) - \mathsf{y}_t \right\|^2 \right].}$$

3. The training target y_t depends on the model:

$$\begin{aligned} &\text{Flow matching:} && \mathsf{y}_t = \dot{\alpha}_t \mathsf{x}_1 + \dot{\sigma}_t \boldsymbol{\epsilon}, \\ &\text{Diffusion models:} && \mathsf{y}_t = \boldsymbol{\epsilon}, \\ &\text{Score matching:} && \mathsf{y}_t = -\boldsymbol{\epsilon} / \sigma_t. \end{aligned}$$

4. The noise schedule, target, and loss weighting may differ, but the regression framework is shared.

Unified training algorithm and key takeaways

1. A common training procedure is:
   - Sample clean data $x_1 \sim p_{\text{data}}$.
   - Sample time t and Gaussian noise $\epsilon \sim \mathcal{N}(0, I)$.
   - Construct $x_t = \alpha_t x_1 + \sigma_t \epsilon$.
   - Compute the model-specific target y_t.
   - Minimize the weighted MSE and update the network parameters.
2. Common aspects:
   - Training uses data samples and sampled noise.
   - Targets can be computed analytically.
   - No marginal probability density evaluation is required.
   - Training uses stochastic gradient optimization.
3. Key differences:
   - The network predicts velocity, noise, or score.
   - Probability paths and loss weights may differ.
   - Generation uses different dynamics or samplers.

Key insight: These methods can share essentially the same training pipeline, with different prediction targets.

A unified view of training algorithms

Flow matching (train velocity)

Target (velocity)

$$\mathbf{u}_t = \dot{\alpha}_t \mathbf{x}_1 + \dot{\sigma}_t \boldsymbol{\epsilon}$$

$(\mathbf{x}_t, t) \rightarrow \mathbf{v}_\theta(\mathbf{x}_t, t)$ predict $\rightarrow \mathbf{u}_t$

Training loss:

$$\mathcal{L}_{\text{FM}} = \mathbb{E}\left[w(t)\|\mathbf{v}_\theta(\mathbf{x}_t, t) - \mathbf{u}_t\|^2\right]$$

Diffusion models (train noise)

Target (noise)

$$\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$$

$(\mathbf{x}_t, t) \rightarrow \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$ predict $\rightarrow \boldsymbol{\epsilon}$

Training loss:

$$\mathcal{L}_{\text{DM}} = \mathbb{E}\left[w(t)\|\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) - \boldsymbol{\epsilon}\|^2\right]$$

Score matching (train score)

Target (score)

$$\mathbf{s}_t = -\boldsymbol{\epsilon}/\sigma_t$$

$(\mathbf{x}_t, t) \rightarrow \mathbf{s}_\theta(\mathbf{x}_t, t)$ predict $\rightarrow \frac{-\boldsymbol{\epsilon}}{\sigma_t}$

Training loss:

$$\mathcal{L}_{\text{SM}} = \mathbb{E}\left[w(t)\|\mathbf{s}_\theta(\mathbf{x}_t, t) + \boldsymbol{\epsilon}/\sigma_t\|^2\right]$$

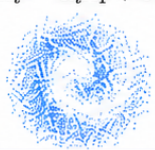
Unified training pipeline

1. Sample clean data $\mathbf{x}_1 \sim p_{\text{data}}$



2. Sample time and noise $t \sim \mathcal{U}(0,1), \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$

3. Construct noisy sample $\mathbf{x}_t = \alpha_t \mathbf{x}_1 + \sigma_t \boldsymbol{\epsilon}$



4. Compute target $\mathbf{y}_t$

Flow matching:	$\mathbf{y}_t = \dot{\alpha}_t \mathbf{x}_1 + \dot{\sigma}_t \boldsymbol{\epsilon}$
Diffusion models:	$\mathbf{y}_t = \boldsymbol{\epsilon}$
Score matching:	$\mathbf{y}_t = -\boldsymbol{\epsilon}/\sigma_t$

5. Train with weighted MSE

$$\mathcal{L} = \mathbb{E}\left[w(t)\|f_\theta(\mathbf{x}_t, t) - \mathbf{y}_t\|^2\right]$$

Update θ by gradient descent.

Comparison of generation methods

1. Flow matching, diffusion models, and score matching generate samples by transforming a simple noise distribution into the data distribution.
2. Their main differences are summarized below:

	Flow matching	Diffusion models	Score matching
Learned quantity	Velocity	Noise	Score
Initial distribution	Gaussian noise	Gaussian noise	Gaussian noise
Generation process	ODE	Reverse diffusion	Reverse SDE / ODE
Stochasticity	Deterministic	Stochastic	Stochastic / deterministic
Numerical method	ODE solver	Denoising steps	SDE / ODE solver

3. The central distinction is how the learned vector field or noise predictor is used to transform noise into data.

Governing equations for generation

1. Flow matching generates samples by integrating a learned velocity field:

$$\boxed{\frac{d\mathsf{x}_t}{dt} = \mathsf{v}_\theta(\mathsf{x}_t, t), \qquad t : 0 \to 1.}$$

2. Diffusion models generate samples by iteratively reversing a noise corruption process:

$$\boxed{\mathsf{x}_{k-1} = \boldsymbol{\mu}_\theta(\mathsf{x}_k, k) + \sigma_k \boldsymbol{\epsilon}_k, \qquad \boldsymbol{\epsilon}_k \sim \mathcal{N}(0, \mathsf{I}).}$$

   Here, $\boldsymbol{\mu}_\theta$ is computed from the learned noise predictor and the noise schedule.

3. Score matching generates samples using the learned score s_θ. For a forward SDE $d\mathsf{x}_t = \mathsf{f}(\mathsf{x}_t, t)\, dt + g(t)\, d\mathsf{W}_t$, the reverse-time SDE is

$$\boxed{d\mathsf{x}_t = \left[\mathsf{f}(\mathsf{x}_t, t) - g^2(t)\mathsf{s}_\theta(\mathsf{x}_t, t)\right] dt + g(t)\, d\overline{\mathsf{W}}_t,}$$

   integrated backward in time.

Unified generation procedure and key takeaways

1. A common generation procedure is:
   - Sample an initial state from Gaussian noise: $\mathrm{x}_{\mathrm{noise}} \sim \mathcal{N}(0, \mathrm{I})$.
   - Evaluate the trained neural network at the current sample and time.
   - Use its prediction to update the sample according to an ODE, SDE, or denoising rule.
   - Repeat until reaching the data endpoint.
2. Common aspects:
   - Generation begins from a simple distribution.
   - A trained network guides the evolution.
   - Sampling typically requires multiple network evaluations.
3. Key differences:
   - Flow matching typically uses deterministic ODE integration.
   - Diffusion models commonly use stochastic reverse denoising, but deterministic samplers exist.
   - Score-based models support both reverse SDE and probability flow ODE sampling.

Key insight: Generation in all three approaches transforms noise into data through a learned, time-dependent dynamical process.

A unified view of generation algorithms

Flow matching (generate with ODE)

Generation dynamics:

$$\frac{d\mathbf{x}_t}{dt} = \mathbf{v}_{\boldsymbol{\theta}}(\mathbf{x}_t, t), \quad t : 0 \to 1.$$

Deterministic ODE integration (e.g., Euler, Heun, or higher-order).

Learned quantity	Velocity $\mathbf{v}_{\boldsymbol{\theta}}(\mathbf{x}, t)$
Initial distribution	Gaussian noise $\mathbf{x}_0 \sim \mathcal{N}(0, \mathbf{I})$
Generation process	Integrate ODE (forward time)
Stochasticity	Deterministic (given initial noise)
Numerical method	ODE solver (e.g., Euler, Heun, RK)

Diffusion models (reverse denoising)

Generation dynamics (discrete):

$$\mathbf{x}_{k-1} = \boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{x}_k, k) + \sigma_k \mathbf{z}_k, \quad \mathbf{z}_k \sim \mathcal{N}(0, \mathbf{I}).$$

Iterative reverse denoising using the learned noise predictor.

Learned quantity	Noise $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}, t)$
Initial distribution	Gaussian noise $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$
Generation process	Reverse diffusion (iterative denoising)
Stochasticity	Stochastic (deterministic samplers exist)
Numerical method	Reverse samplers (e.g., DDPM, DDIM)

Score matching (reverse SDE / ODE)

Generation dynamics (reverse-time SDE):

$$d\mathbf{x}_t = \left[f(\mathbf{x}_t, t) - g^2(t) \boldsymbol{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) \right] dt + g(t) d\bar{\boldsymbol{W}}_t, \quad t : 1 \to 0.$$

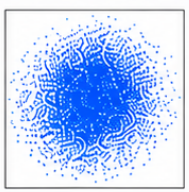
Stochastic SDE integration (e.g., Euler–Maruyama) or deterministic probability flow ODE.

Learned quantity	Score $\boldsymbol{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)$
Initial distribution	Gaussian noise $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$
Generation process	Reverse SDE / probability flow ODE
Stochasticity	Stochastic / deterministic
Numerical method	SDE: Euler–Maruyama; ODE: Runge–Kutta

Unified generation pipeline

1 Sample initial noise

$\mathbf{x}_{\text{init}} \sim \mathcal{N}(0, \mathbf{I})$

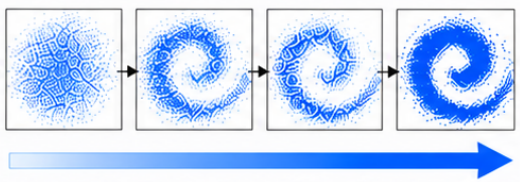


2 Iteratively update samples

Use the trained model according to the chosen dynamics:

- Flow matching: ODE (forward)
- Diffusion models: reverse denoising
- Score matching: reverse SDE / ODE

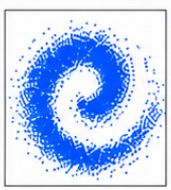
3 Intermediate sample distributions



Noise → Intermediate sample distributions → Data

4 Generated samples

$\mathbf{x}_{\text{final}} \sim p_{\text{model}}$



5 Key takeaways

- Start from a simple distribution (Gaussian noise).
- Use a trained neural network to guide the evolution.
- Sampling typically requires multiple network evaluations.
- Flow matching uses deterministic ODE integration.
- Diffusion models commonly use stochastic reverse denoising, but deterministic samplers exist.
- Score-based models support both reverse SDE and probability flow ODE.
- All three methods use trained neural networks to guide generation, but learn different quantities.

Putting everything together

1. The following three equations provide a useful summary of the relationships.
2. Flow matching: deterministic transport

$$\frac{dX_t}{dt} = v_t(X_t)$$

   The velocity field transports probability mass.
3. Score matching: density geometry

$$s_t(x) = \nabla \log p_t(x)$$

   The score describes the local geometry of the log-density.
4. Diffusion: stochastic dynamics

$$dX_t = f_t(X_t)\, dt + g_t\, dW_t$$

   Brownian motion introduces stochastic fluctuations.

Putting everything together

1. The bridge between them is the evolution of probability distributions.
2. For deterministic flows, that evolution is governed by the continuity equation.
3. For stochastic diffusion, it is governed by the Fokker-Planck equation.
4. The score function allows us to rewrite the Fokker-Planck equation in continuity-equation form.
5. This is why these apparently different generative modeling frameworks can be closely related mathematically.

Outline

1 Introduction

2 Flow Matching

3 Diffusion Models

4 Score Matching

5 Connections

6 Frontiers
- Fast Sampling and Consistency Models
- Stochastic Interpolants and Schrödinger Bridges
- Generative Modeling for Scientific Data

7 Latent Variable Models

8 Variational Inference

Why is generative sampling expensive?

1. Diffusion and flow models generate samples by solving an ODE or SDE, typically requiring multiple neural network evaluations.
2. For an ODE-based generative model,

$$\frac{dz_t}{dt} = v_\theta(z_t, t),$$

   numerical integration approximates a continuous trajectory using discrete time steps.
3. For example, the Euler method gives

$$z_{t+\Delta t} = z_t + \Delta t\, v_\theta(z_t, t).$$

4. Smaller time steps generally improve numerical accuracy but require more evaluations of v_θ.
5. A key efficiency measure is the number of function evaluations (NFE), often approximately equal to the number of neural network forward passes.

Numerical accuracy versus sampling cost

1. Consider an ODE

$$\frac{dz_t}{dt} = v_\theta(z_t, t), \qquad t \in [0, 1].$$

2. With N uniform steps, the step size is $\Delta t = 1/N$.
3. For sufficiently smooth velocity fields:
   - Euler method: global error $\mathcal{O}(\Delta t)$.
   - Heun method: global error $\mathcal{O}(\Delta t^2)$.
   - Higher-order solvers can achieve better accuracy with fewer steps.
4. However, higher-order solvers generally require more function evaluations per step.
5. The practical objective is to achieve good sample quality with a small NFE.

Sources of sampling error

1. Sampling error can arise from several sources:
   - Model error: the learned velocity or score differs from the ideal field.
   - Discretization error: numerical integration approximates continuous dynamics.
   - Distillation error: a student model imperfectly reproduces a teacher's generation process.
2. Let x^* denote an ideal sample associated with the same initial noise, x_{teacher} the teacher's output, and x_{student} the student's output.
3. By the triangle inequality,

$$\|x_{\text{student}} - x^*\| \leq \|x_{\text{student}} - x_{\text{teacher}}\| + \|x_{\text{teacher}} - x^*\|.$$

4. The first term measures student–teacher mismatch; the second contains the teacher's modeling and numerical errors.
5. Faster sampling must balance computational cost against these sources of error.

Distillation for fast generation

1. Distillation transfers the behavior of an accurate, multi-step teacher model into a faster student model.
2. Let $G_{\text{teacher}}(z_T)$ denote the teacher's final sample from initial noise z_T.
3. A student generator G_ϕ can be trained using

$$\mathcal{L}_{\text{distill}}(\phi) = \mathbb{E}_{z_T \sim p_T}\left[\left\| G_\phi(z_T) - G_{\text{teacher}}(z_T)\right\|^2\right].$$

4. The student may use fewer numerical steps or directly predict the final sample.
5. Distillation reduces inference cost, but requires additional training and can inherit teacher errors.
6. Examples include progressive distillation and consistency distillation.

Consistency models

1. Consider the probability flow ODE of a diffusion model:

$$\frac{dz_t}{dt} = v(z_t, t).$$

2. States along the same ODE trajectory correspond to the same final clean sample.
3. A consistency model learns a function $f_\theta(z, t)$ that maps any state on a trajectory to its endpoint at a small time $\varepsilon > 0$.
4. The consistency property is

$$f_\theta(z_t, t) = f_\theta(z_s, s), \qquad z_t, z_s \text{ on the same trajectory.}$$

5. The boundary condition is

$$f_\theta(z_\varepsilon, \varepsilon) = z_\varepsilon.$$

6. Once trained, the model can generate samples using one or a few neural network evaluations.

Training consistency models

1. Choose adjacent time points $t_{n+1} > t_n \geq \varepsilon$ and sample a noisy state $z_{t_{n+1}}$.
2. Use a numerical ODE solver with a pretrained teacher to estimate the earlier state:
$$\widehat{z}_{t_n} = \text{Solver}_{\text{teacher}}(z_{t_{n+1}}, t_{n+1}, t_n).$$
3. Enforce consistency between the two states:
$$\mathcal{L}_{\text{CM}}(\theta) = \mathbb{E}\Big[d\big(f_\theta(z_{t_{n+1}}, t_{n+1}), f_{\bar{\theta}}(\widehat{z}_{t_n}, t_n)\big)\Big].$$
4. Here d is a distance measure and $\bar{\theta}$ denotes parameters of a stop-gradient target network.
5. The model learns approximately invariant predictions along teacher ODE trajectories.
6. Consistency models can also be trained without a pretrained teacher using consistency training.

One-step and few-step generation

1. For one-step generation, sample from the terminal noise distribution:

$$z_T \sim p_T(z), \qquad \hat{x} = f_\theta(z_T, T).$$

2. This requires only one evaluation of the consistency function.
3. Few-step generation alternates consistency prediction and controlled noise injection:

$$\hat{x}_k = f_\theta(z_{t_k}, t_k),$$

$$z_{t_{k-1}} = \hat{x}_k + \sigma_{t_{k-1}} \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I).$$

4. The second equation is illustrative of variance-exploding noise schedules; other schedules use different noise injection rules.
5. Additional steps can improve sample quality, but increase NFE.

Comparing fast sampling approaches

1. Several approaches improve generation efficiency:
   - Better numerical solvers reduce discretization error per function evaluation.
   - Distillation transfers multi-step generation into a faster student.
   - Consistency models learn direct mappings from noisy states to trajectory endpoints.
   - Rectified flow seeks straighter trajectories, making coarse numerical integration more accurate.
2. These approaches address different aspects of the sampling problem and can be complementary.
3. The main tradeoff is between inference cost, training cost, approximation error, and sample quality.
4. Fast generation connects continuous-time generative modeling with practical deployment.

Stochastic interpolants: a unified perspective

1. Flow matching and diffusion models construct paths of distributions p_t from p_0 to $p_1 = p_{\text{data}}$.
2. A stochastic interpolant constructs random states X_t with $X_0 \sim p_0$ and $X_1 \sim p_1$.
3. The same marginal path p_t may be realized by different dynamics:
   - A deterministic ODE transporting probability mass.
   - A stochastic SDE combining drift and diffusion.
4. Schrödinger bridges impose an additional principle: choose a path measure closest to a reference stochastic process.

Constructing a stochastic interpolant

1. Sample $(X_0, X_1) \sim \pi$, where π couples p_0 and p_1, and independently sample $\epsilon \sim \mathcal{N}(0, I)$.
2. Define, for $t \in [0, 1]$,

$$X_t = \alpha(t)X_0 + \beta(t)X_1 + \gamma(t)\epsilon.$$

3. Choose endpoint conditions

$$\begin{aligned} \alpha(0) = 1, \quad \alpha(1) = 0, \\ \beta(0) = 0, \quad \beta(1) = 1, \\ \gamma(0) = \gamma(1) = 0. \end{aligned}$$

4. The coupling and interpolation schedules determine the intermediate marginals p_t; the endpoint distributions alone do not.

Special cases and modeling choices

1. Linear flow matching uses

$$\alpha(t) = 1 - t, \quad \beta(t) = t, \quad \gamma(t) = 0,$$

   so $X_t = (1 - t)X_0 + tX_1$.
2. A nonzero $\gamma(t)$ adds Gaussian randomness at intermediate times while leaving the endpoints unchanged.
3. Different endpoint couplings π produce different probability paths, even with identical schedules.
4. The interpolant specifies the random variables X_t and their marginals; it does not uniquely specify the Markov dynamics used for generation.

Conditional and marginal velocity fields

1. Differentiate each sampled interpolation path:

$$\dot{X}_t = \dot{\alpha}(t)X_0 + \dot{\beta}(t)X_1 + \dot{\gamma}(t)\epsilon.$$

2. Its marginal velocity is the conditional expectation

$$v_t(x) = \mathbb{E}[\dot{X}_t \mid X_t = x].$$

3. Train a neural vector field using the sampled target:

$$\mathcal{L}_{\text{velocity}}(\theta) = \mathbb{E}\left[\|v_\theta(X_t, t) - \dot{X}_t\|^2\right].$$

4. By least-squares regression, the population minimizer is $v_t(x)$; no evaluation of $p_t(x)$ is required.

Why the marginal velocity conserves probability

1. For a smooth, compactly supported test function φ, differentiate its expectation:

$$\begin{aligned}\frac{d}{dt}\mathbb{E}[\varphi(X_t)] &= \mathbb{E}[\nabla\varphi(X_t)\cdot\dot{X}_t] \\ &= \mathbb{E}[\nabla\varphi(X_t)\cdot v_t(X_t)] \\ &= \int \nabla\varphi(x)\cdot v_t(x)p_t(x)\,dx.\end{aligned}$$

2. Integration by parts gives

$$\int \varphi(x)\,\partial_t p_t(x)\,dx = -\int \varphi(x)\,\nabla\cdot(p_t v_t)(x)\,dx.$$

3. Therefore, in the weak sense,

$$\partial_t p_t + \nabla\cdot(p_t v_t) = 0.$$

Learning the score from interpolant samples

1. Assume $\gamma(t) > 0$ for $0 < t < 1$. Conditioned on endpoints,

$$p_t(x \mid x_0, x_1) = \mathcal{N}\left(x; \alpha x_0 + \beta x_1, \gamma^2 I\right).$$

2. Its conditional score is

$$\nabla_x \log p_t(x \mid x_0, x_1) = -\frac{x - \alpha x_0 - \beta x_1}{\gamma^2} = -\frac{\epsilon}{\gamma}.$$

3. Averaging conditional scores given $X_t = x$ yields

$$s_t(x) := \nabla_x \log p_t(x) = -\frac{1}{\gamma(t)} \mathbb{E}[\epsilon \mid X_t = x].$$

Score regression and endpoint caveats

1. A score network can be trained by denoising regression:

$$\mathcal{L}_{\text{score}}(\theta) = \mathbb{E}\left[\left\| s_\theta(X_t, t) + \frac{\epsilon}{\gamma(t)} \right\|^2\right].$$

2. Its population minimizer is $s_t(x) = \nabla_x \log p_t(x)$, by conditional expectation.
3. The formula requires $\gamma(t) > 0$ and suitable differentiability of the marginal density.
4. Since $\gamma(0) = \gamma(1) = 0$, the target may become singular near endpoints; practical training uses interior times and suitable time weighting.

One probability path, multiple dynamics

1. The marginal density satisfies

$$\partial_t p_t = -\nabla \cdot (p_t v_t).$$

2. An ODE realizes the path:

$$dX_t = v_t(X_t)\, dt.$$

3. For any $\eta(t) \geq 0$, an SDE can realize the same marginals:

$$dX_t = \left[v_t(X_t) + \eta(t) s_t(X_t)\right] dt + \sqrt{2\eta(t)}\, dW_t.$$

4. This equality concerns one-time marginals p_t, not equality of individual trajectories or path measures.

Why the score correction cancels diffusion

1. For $dX_t = b_t(X_t)dt + \sqrt{2\eta(t)}dW_t$, the Fokker–Planck equation is

$$\partial_t p_t = -\nabla \cdot (p_t b_t) + \eta(t)\Delta p_t.$$

2. Set $b_t = v_t + \eta(t)s_t$ and use $p_t s_t = \nabla p_t$:

$$\begin{aligned}\partial_t p_t &= -\nabla \cdot (p_t v_t + \eta p_t s_t) + \eta \Delta p_t \\ &= -\nabla \cdot (p_t v_t) - \eta \Delta p_t + \eta \Delta p_t \\ &= -\nabla \cdot (p_t v_t).\end{aligned}$$

3. The additional drift precisely offsets the probability flux caused by Brownian diffusion.

Probability currents: the physical interpretation

1. Write the Fokker–Planck equation as a conservation law:

$$\partial_t p_t + \nabla \cdot J_t = 0, \qquad J_t = p_t b_t - \eta \nabla p_t.$$

2. The current contains two contributions:
   - Drift current $p_t b_t$.
   - Diffusive current $-\eta \nabla p_t$.
3. With $b_t = v_t + \eta s_t$,

$$J_t = p_t(v_t + \eta s_t) - \eta \nabla p_t = p_t v_t.$$

4. Thus the ODE and corrected SDE transport the same probability current, despite different sample trajectories.

Why Schrödinger bridges?

1. A stochastic interpolant supplies a possible probability path, but generally does not select an optimal path.
2. Fix a reference stochastic process with path measure R; for example, Brownian motion with a prescribed initial distribution.
3. Seek a new process with the desired endpoint marginals while changing the reference dynamics as little as possible.
4. The notion of closeness is relative entropy between probability measures on complete trajectories, not just between endpoint distributions.

Schrödinger bridges as path-space optimization

1. Let P and R denote probability measures on trajectories $X_{[0,1]}$.
2. The dynamic Schrödinger bridge problem is

$$\begin{aligned} \min_{P} \quad & D_{\mathrm{KL}}(P\|R) \\ \text{subject to} \quad & P_0 = p_0, \qquad P_1 = p_1. \end{aligned}$$

3. P_0 and P_1 are the endpoint marginals induced by P.
4. Unlike an arbitrary interpolant, the optimizer is selected by a global path-space variational principle.
5. Existence and uniqueness require suitable compatibility and absolute-continuity assumptions.

From path-space KL to stochastic control

1. Choose a Brownian reference with $X_0 \sim p_0$:

$$dX_t = \sqrt{2\varepsilon}\, dW_t, \qquad \varepsilon > 0.$$

2. Consider controlled processes with the same initial law:

$$dX_t = u_t(X_t)\, dt + \sqrt{2\varepsilon}\, dW_t.$$

3. Under Girsanov's theorem and regularity assumptions,

$$D_{\mathrm{KL}}(P^u \| R) = \frac{1}{4\varepsilon} \mathbb{E}_{P^u} \left[\int_0^1 \| u_t(X_t) \|^2 \, dt \right].$$

4. Therefore the bridge minimizes expected control energy subject to $X_1 \sim p_1$ (and $X_0 \sim p_0$).

Schrödinger potentials and the optimal drift

1. Under suitable conditions, the optimal path measure has the endpoint-tilted form
$$\frac{dP^*}{dR} = f(X_0)g(X_1),$$
where positive functions f, g enforce the two endpoint marginals.
2. Define the backward potential
$$g_t(x) = \mathbb{E}_R[g(X_1) \mid X_t = x].$$
3. For a Brownian reference, the optimally controlled forward drift is
$$u_t^*(x) = 2\varepsilon \nabla_x \log g_t(x).$$
4. The potential is obtained by solving the endpoint matching problem; it is not determined by an arbitrary interpolation schedule.

Schrödinger bridges and entropic optimal transport

1. Let R_{01} be the joint endpoint distribution of the reference process.
2. The static Schrödinger problem is

$$\min_{\pi \in \Pi(p_0, p_1)} D_{\mathrm{KL}}(\pi \| R_{01}).$$

3. For Brownian motion, the transition kernel has Gaussian form

$$r_\varepsilon(x_1 \mid x_0) \propto \exp\left(-\frac{\|x_1 - x_0\|^2}{4\varepsilon}\right).$$

4. Consequently the objective combines a quadratic transport cost with an entropy term, up to terms fixed by the marginals.
5. The optimal path measure also uses the reference's conditional bridges between endpoint pairs.

Small-noise limit and classical optimal transport

1. In the Brownian reference, ε controls the magnitude of random fluctuations.
2. The endpoint transition penalty contains

$$\frac{\|x_1 - x_0\|^2}{4\varepsilon}.$$

3. As $\varepsilon \to 0$, entropy-regularized transport approaches quadratic-cost optimal transport under appropriate assumptions.
4. After a suitable rescaling, the optimal stochastic transport interpolations converge to displacement interpolations in the classical optimal-transport problem.
5. This is a limiting relationship, not an assertion that every stochastic interpolant is an optimal bridge.

Stochastic interpolants versus Schrödinger bridges

1. Stochastic interpolants choose a coupling and interpolation schedules to construct X_t and its marginals.
2. Schrödinger bridges choose a path measure by minimizing $D_{\mathrm{KL}}(P\|R)$ subject to endpoint constraints.
3. Both can be studied using velocity, score, continuity, and Fokker–Planck equations.
4. The distinction is between constructing a valid path and optimizing a path relative to a specified stochastic reference.
5. Simulation-free score and flow regression can also be used in methods designed to approximate Schrödinger bridges.

Connecting flow matching, diffusion, and transport

1. Flow matching learns a velocity v_t that transports prescribed marginals by an ODE.
2. Stochastic interpolants permit learning both v_t and the score s_t from sampled conditional paths.
3. Score-corrected SDEs can reproduce the same marginals with different pathwise randomness.
4. Schrödinger bridges select a stochastic transport process using path-space relative entropy.
5. Optimal transport emerges as a small-noise limit of the Brownian Schrödinger bridge under suitable assumptions.

Generative modeling for scientific data

1. Scientific data include molecular configurations, material structures, and spatial fields such as fluid velocity.
2. Beyond distributional fidelity, generated samples may need to satisfy:
   - Physical constraints, such as incompressibility or boundary conditions.
   - Symmetries, such as rotations, translations, and particle permutations.
   - Conservation or balance laws for mass, momentum, and energy.
3. A conditional scientific generator models

$$p_\theta(\mathrm{x} \mid \mathrm{c}),$$

   where c may encode initial conditions, parameters, or measurements.
4. The goal is to generate plausible states that are also physically admissible.

Symmetries and physical constraints

1. Let g be a symmetry transformation acting on physical states. An equivariant generative map satisfies

$$G_\theta(g \cdot \mathrm{x}) = g \cdot G_\theta(\mathrm{x}).$$

2. Examples of structure-aware modeling include:
   - Rotation-equivariant models for molecular coordinates and vector fields.
   - Permutation-equivariant models for sets of identical particles.
   - Constraint-preserving parameterizations or projection operators.
3. A soft constraint can be imposed through a penalty:

$$\mathcal{L}(\theta) = \mathcal{L}_{\mathrm{gen}}(\theta) + \lambda\, \mathbb{E}\left[\|\mathcal{C}(\mathrm{x}_{\mathrm{gen}})\|^2\right].$$

4. Soft penalties encourage compliance; exact parameterizations or projections can enforce selected constraints more directly.

Example: incompressible turbulent flows

1. A physical state is a velocity field $\mathsf{u}(\mathsf{r}, \tau)$, where r is position and τ is physical time.
2. For incompressible flow,
$$\nabla_{\mathsf{r}} \cdot \mathsf{u} = 0.$$
3. Its dynamics obey the incompressible Navier–Stokes equations:
$$\partial_\tau \mathsf{u} + (\mathsf{u} \cdot \nabla_{\mathsf{r}})\mathsf{u} = -\nabla_{\mathsf{r}} p + \nu \Delta_{\mathsf{r}} \mathsf{u} + \mathsf{f},$$
$$\nabla_{\mathsf{r}} \cdot \mathsf{u} = 0.$$
4. The kinetic energy $E(\tau) = \frac{1}{2} \int_\Omega \|\mathsf{u}\|^2 \, d\mathsf{r}$ obeys a balance involving forcing, dissipation, and boundary fluxes.
5. Hence physical validity is not equivalent to matching pixelwise errors or marginal distributions alone.

Physics-aware stochastic interpolants

1. Encode physical states as vectors: x_0 is a source state and x_1 is a target state, with x_t at generative time $t \in [0, 1]$.
2. A stochastic interpolant takes the form

$$\mathsf{x}_t = \alpha(t)\mathsf{x}_0 + \beta(t)\mathsf{x}_1 + \gamma(t)\boldsymbol{\epsilon}.$$

3. Physical structure can enter through:
   - The choice of source–target coupling or conditioning.
   - Learnable interpolation coefficients and physics-aware losses.
   - A projection of generated states onto an admissible subspace.
4. For an incompressible velocity field, a Helmholtz-type projection is

$$\mathcal{P}\mathsf{u} = \mathsf{u} - \nabla_{\mathsf{r}}\phi, \qquad \Delta_{\mathsf{r}}\phi = \nabla_{\mathsf{r}} \cdot \mathsf{u},$$

with appropriate boundary conditions.

Evaluating scientific generative models

1. Distributional accuracy: compare ensembles, correlations, and relevant physical statistics.
2. Physical admissibility: measure constraint violations, such as

$$\mathcal{E}_{\mathrm{div}} = \mathbb{E}\left[\|\nabla_{\mathsf{r}} \cdot \mathsf{u}_{\mathrm{gen}}\|^2\right].$$

3. Multiscale fidelity: compare turbulent kinetic-energy spectra $E(k)$ across wavenumbers k.
4. Dynamical validity: evaluate long-rollout stability, energy balance, and forecast uncertainty.
5. Conditional fidelity: verify agreement with observed or prescribed initial and boundary data.
6. Scientific quality requires both statistical realism and physical consistency; neither alone is sufficient.

Outline

1 Introduction

2 Flow Matching

3 Diffusion Models

4 Score Matching

5 Connections

6 Frontiers

7 Latent Variable Models
- Latent Variable Models and Inference
- Maximum Likelihood and Inference Challenges
- Complete-Data and Marginal Likelihood
- Gaussian Mixtures and Expectation-Maximization

8 Variational Inference

A unified view of generative modeling

1. Generative modeling aims to learn a distribution $p_\theta(\mathrm{x})$ that approximates the data distribution $p_{\mathrm{data}}(\mathrm{x})$.
2. Different models describe generation in different ways:
   - Flow matching: transport samples using a learned velocity field.
   - Diffusion models: generate samples by reversing a noise-corruption process.
   - Score matching: learn the score function (gradient of the log-probability density) to guide generative sampling.
3. In general, latent variable models generate observations from hidden random variables.
4. These perspectives are closely related: continuous-time generative processes can also be interpreted as latent variable models.
5. This connection provides a unified view of generation, likelihood, and inference.

Flow matching as latent variable models

1. In flow matching, generation starts from a simple distribution:

$$\mathrm{x}_0 \sim p_0(\mathrm{x}) = \mathcal{N}(0, \mathrm{I}).$$

2. A learned velocity field defines an ODE:

$$\frac{d\mathrm{x}_t}{dt} = \mathrm{v}_\theta(\mathrm{x}_t, t).$$

3. Solving the ODE from $t = 0$ to $t = 1$ defines a transformation:

$$\mathrm{x}_1 = g_\theta(\mathrm{x}_0).$$

4. Thus, the initial random variable x_0 plays the role of a latent variable, while x_1 is the generated observation.
5. Flow matching can therefore be interpreted as learning a latent-to-observation transformation.

Diffusion models and hidden trajectories

1. Diffusion models introduce a sequence of noisy latent variables:

$$x \to z_1 \to z_2 \to \cdots \to z_T.$$

2. The forward diffusion process is

$$q(z_{1:T} \mid x) = q(z_1 \mid x) \prod_{t=2}^{T} q(z_t \mid z_{t-1}).$$

3. The generative model reverses this process:

$$p_\theta(x, z_{1:T}) = p(z_T) \left[\prod_{t=2}^{T} p_\theta(z_{t-1} \mid z_t) \right] p_\theta(x \mid z_1).$$

4. The observed-data likelihood is obtained by marginalizing the latent variables:

$$p_\theta(x) = \int p_\theta(x, z_{1:T}) \, dz_{1:T}.$$

5. Thus, diffusion models are hierarchical latent variable models with a sequence of hidden noisy states.

Score matching and hidden trajectories

1. Score-based generative models introduce a continuous-time sequence of noisy latent states:

$$x \to z_0 \to z_t \to z_T, \qquad z_0 = x.$$

2. The forward stochastic process is described by an SDE:

$$dz_t = f(z_t, t)\, dt + g(t)\, dW_t.$$

3. Score matching learns the score of the time-dependent marginal distribution:

$$s_\theta(z_t, t) \approx \nabla_{z_t} \log p_t(z_t).$$

4. The learned score defines the reverse-time SDE:

$$dz_t = \left[f(z_t, t) - g^2(t) s_\theta(z_t, t)\right] dt + g(t)\, d\overline{W}_t,$$

integrated from T to 0, with $d\overline{W}_t$ denoting reverse-time Brownian noise.

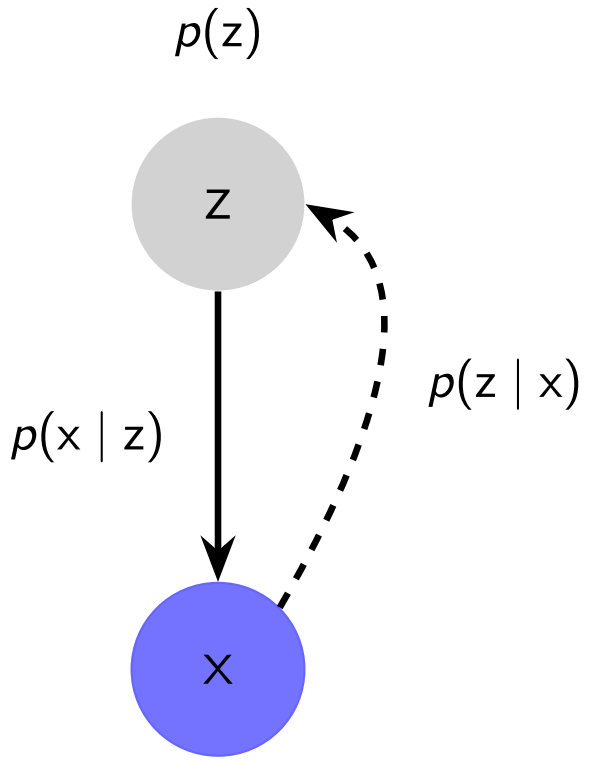
5. Thus, score-based generation can be viewed as a latent stochastic process whose reverse dynamics are determined by learned score fields.

Latent variable models

1. A latent variable model defines a distribution over observations x by using a (vector) latent variable z and specifying:
   - The prior distribution $p(z)$ for the latent variable
   - The likelihood $p(x \mid z)$ that connects the latent variable to the observation
2. The prior and the likelihood define the joint distribution

$$p(x, z) = p(x \mid z)p(z)$$

3. Two central questions are: how likely is an observation x, and which latent values plausibly explain it? These correspond to the marginal likelihood $p_\theta(x)$ and posterior $p_\theta(z \mid x)$.



Why introduce latent variables?

1. *Simpler explanations:* a high-dimensional observation may be controlled by a smaller collection of underlying factors, such as object identity, pose, or physical state.
2. *Dependence:* observed coordinates can be dependent even if they are conditionally independent given z:

$$p_\theta(\mathrm{x} \mid \mathrm{z}) = \prod_{j=1}^{d} p_\theta(x_j \mid \mathrm{z}), \qquad p_\theta(\mathrm{x}) = \int \prod_{j=1}^{d} p_\theta(x_j \mid \mathrm{z}) p_\theta(\mathrm{z})\, d\mathrm{z}.$$

   The marginal distribution $p_\theta(\mathrm{x})$ need not factorize.
3. *Representation:* the posterior over z provides a data-dependent representation, including uncertainty about possible explanations.
4. *Generation:* sample a simple latent variable and then sample an observation from the conditional distribution.
5. A latent variable is not automatically interpretable or low-dimensional; those properties depend on modeling assumptions and learning.

Generation and inference

1. *Generation* follows the model's causal sampling direction:

$$\boxed{z \sim p_\theta(z), \qquad x \sim p_\theta(x \mid z).}$$

   The prior describes plausible latent states; the likelihood describes observations conditioned on them.

2. *Inference* runs in the reverse informational direction: after observing x, determine which z values remain plausible.

$$\boxed{p_\theta(z \mid x) = \frac{p_\theta(x \mid z)p_\theta(z)}{p_\theta(x)}.}$$

3. Generation needs sampling from the prior and likelihood; inference needs conditioning on an observation and generally involves normalization.
4. The inverse is probabilistic: several latent configurations may explain the same observation, particularly when the likelihood is noisy or many-to-one.

Posterior inference and marginal likelihood

1. Bayes' rule gives the posterior:

$$p_\theta(z \mid x) = \frac{p_\theta(x, z)}{p_\theta(x)} = \frac{p_\theta(x \mid z)p_\theta(z)}{\int p_\theta(x \mid z')p_\theta(z')\, dz'}.$$

2. The denominator is the *marginal likelihood*, also called the *evidence*:

$$\boxed{p_\theta(x) = \int p_\theta(x, z)\, dz.}$$

   It measures how well the model explains an observation after considering all possible latent explanations.

3. Marginalization is a weighted average of likelihoods over the prior, not a choice of only the most likely latent configuration.

4. Although the joint $p_\theta(x, z)$ may be easy to evaluate, integrating over a high-dimensional z can make both the evidence and posterior difficult to compute.

Inference is the inverse of generation

1. The same joint distribution has two factorizations:

$$\boxed{p_\theta(x, z) = p_\theta(x \mid z)p_\theta(z) = p_\theta(z \mid x)p_\theta(x).}$$

2. Thus, two idealized sampling procedures produce the same joint law:

$$\begin{aligned} &\text{Forward: } z \sim p_\theta(z), \quad x \sim p_\theta(x \mid z), \\ &\text{Reverse: } x \sim p_\theta(x), \quad z \sim p_\theta(z \mid x). \end{aligned}$$

3. The reverse procedure is a conceptual identity, not necessarily an efficient algorithm: sampling $p_\theta(x)$ or evaluating the posterior may be hard.
4. A deterministic inverse $z = f^{-1}(x)$ is a special case; a general latent variable model instead yields a distribution over explanations.

Why is inference important?

1. *Explaining data:* the posterior identifies latent configurations compatible with an observation and quantifies ambiguity.
2. *Prediction:* latent uncertainty can be propagated into predictions of unobserved quantities y:

$$p_\theta(\mathrm{y} \mid \mathrm{x}) = \int p_\theta(\mathrm{y} \mid \mathrm{z}, \mathrm{x}) p_\theta(\mathrm{z} \mid \mathrm{x})\, d\mathrm{z}.$$

3. *Learning:* because the latent values are missing, their posterior probabilities determine how strongly different latent configurations contribute to the parameter update.
4. *Connection to earlier models:* conditional-to-marginal averaging also appeared in flow matching and score matching. Here the hidden variable is z, and posterior averaging appears in likelihood gradients.

Maximum likelihood learning

1. For independent observations $\mathcal{D} = \{x_1, \ldots, x_N\}$, maximum likelihood chooses parameters that make the observed data probable:

$$\boxed{\theta_{\mathrm{ML}} = \arg\max_{\theta} \sum_{n=1}^{N} \log p_\theta(x_n).}$$

2. In a latent variable model, each term is a log marginal likelihood:

$$\log p_\theta(x_n) = \log \int p_\theta(x_n, z_n)\, dz_n.$$

3. The difficulty is not merely optimizing θ: evaluating the objective or its gradient can require a high-dimensional integral for every data point.
4. We therefore distinguish *model parameters* θ from *latent variables* z_n. The former are learned across the dataset; the latter describe individual observations.

Gradient of the marginal log-likelihood

1. Differentiate the marginal likelihood, assuming differentiation and integration can be interchanged:

$$\begin{aligned}\nabla_\theta \log p_\theta(\mathrm{x}) &= \frac{\int \nabla_\theta p_\theta(\mathrm{x},\mathrm{z})\, d\mathrm{z}}{p_\theta(\mathrm{x})} \\ &= \int \frac{p_\theta(\mathrm{x},\mathrm{z})}{p_\theta(\mathrm{x})} \nabla_\theta \log p_\theta(\mathrm{x},\mathrm{z})\, d\mathrm{z} \\ &= \boxed{\mathbb{E}_{p_\theta(\mathrm{z}|\mathrm{x})}[\nabla_\theta \log p_\theta(\mathrm{x},\mathrm{z})]}.\end{aligned}$$

2. This is often called *Fisher's identity*: the observed-data score is the posterior average of the complete-data score.
3. Interpretation: latent configurations that better explain x receive greater weight in the parameter update (posterior credit assignment).
4. It also exposes the bottleneck: computing the likelihood gradient generally requires posterior expectations.

Why exact inference is hard

1. Continuous latent variables require integrals such as

$$p_\theta(\mathrm{x}) = \int_{\mathbb{R}^k} p_\theta(\mathrm{x}, \mathrm{z})\, d\mathrm{z}.$$

   For nonlinear neural-network likelihoods, these integrals rarely have closed forms.

2. Discrete latent variables require sums; with k binary latents, a direct sum can contain 2^k terms.
3. Exact inference is possible in important structured cases, including conjugate models, linear-Gaussian systems, and certain graphical models.
4. Mixture models with a small number of components admit tractable posterior responsibilities. Invertible deterministic maps admit direct inversion, subject to their assumptions.
5. *Key distinction:* easy joint-density evaluation does not imply easy marginal likelihood or posterior inference.

Strategies for handling intractable inference

1. *Choose tractable structure:* design a model with analytic marginalization, efficient message passing, or an invertible transformation.
2. *Approximate inference:* retain a flexible model and approximate the posterior or its expectations numerically.
3. Two broad approaches to approximation are:
   - Monte Carlo methods, which approximate expectations using samples.
   - Variational methods, which fit a tractable distribution to the posterior.
4. There is a tradeoff among model expressiveness, inference accuracy, computational cost, and ease of optimization. Tractability is a property of the combination of model and inference task, not simply the presence of latent variables.

Invertible models as a tractable special case

1. Let $z \sim p_0(z)$ and generate $x = f_\theta(z)$, where $f_\theta : \mathbb{R}^d \to \mathbb{R}^d$ is an invertible differentiable map.
2. Inference is deterministic: $z = f_\theta^{-1}(x)$. The change-of-variables formula gives

$$\boxed{p_\theta(x) = p_0(f_\theta^{-1}(x)) \left|\det Df_\theta^{-1}(x)\right|.}$$

3. This allows exact density evaluation when inversion and the Jacobian determinant are computationally tractable.
4. Unlike a general stochastic decoder $p_\theta(x \mid z)$, a deterministic bijection has a single latent preimage for each observation.
5. This connects to earlier discussions of normalizing flows; exact likelihood computation depends on the architecture, not on invertibility alone.

Limitations and scope of invertible models

1. A standard smooth bijection between Euclidean latent and observation spaces requires matching dimensions. Lower-dimensional latent manifolds require modified constructions.
2. A bijective continuous map does not directly describe arbitrary discrete observations; dequantization or other likelihood constructions may be needed.
3. Efficient Jacobian determinants and inverses constrain architectural choices, even though expressive normalizing flows exist.
4. Invertibility is useful when exact density evaluation is central, but a stochastic latent-variable model may better represent ambiguity, observation noise, or domain-specific structure.
5. *Connection to flow matching:* an ODE flow can define an invertible map under suitable regularity. Flow matching supplies a way to train its velocity field, but evaluating exact likelihood still requires tracking density change.

Why accept intractable models?

1. Scientific and structured models often prioritize meaningful latent factors, physical constraints, or uncertainty over easy marginalization.
2. A flexible decoder can encode domain knowledge, for example

$$p_\theta(\mathrm{x}, \mathrm{z}) = p_\theta(\mathrm{z}) \prod_{j=1}^{d} p_\theta(x_j \mid \mathrm{z}),$$

   while the induced marginal over x remains highly dependent and potentially complex.
3. The latent dimension can differ from the observation dimension; stochasticity can represent multiple possible observations from one latent state.
4. The resulting inference problem may be difficult, but approximate inference makes learning and uncertainty estimation possible.
5. The goal is not intractability itself: it is to choose a useful model and then find a practical way to reason about its hidden variables.

Approximate inference: Monte Carlo and variational

1. *Markov chain Monte Carlo (MCMC):* construct a Markov chain with stationary distribution $p_\theta(z \mid x)$ and approximate posterior expectations using samples.

$$\mathbb{E}_{p_\theta(z|x)}[h(z)] \approx \frac{1}{M} \sum_{m=1}^{M} h(z^{(m)}).$$

   Under suitable assumptions, the approximation improves as the chain runs longer, but mixing and convergence can be difficult.

2. *Variational inference (VI):* choose a tractable family $q_\phi(z \mid x)$ and optimize its parameters to approximate the posterior.
3. MCMC primarily trades computation for sampling accuracy; VI trades optimization efficiency for restrictions imposed by its approximation family and objective.
4. Neither method is universally superior: posterior geometry, dimensionality, and computational budget matter.

Complete-data likelihood for a dataset

1. Consider independent observations and their associated latent variables:
$$\mathcal{D} = \{\mathsf{x}_n\}_{n=1}^N, \qquad Z = \{\mathsf{z}_n\}_{n=1}^N.$$
2. If the latent variables were observed, the complete-data likelihood would factorize:
$$\begin{aligned} p_\theta(\mathcal{D}, Z) &= \prod_{n=1}^N p_\theta(\mathsf{x}_n, \mathsf{z}_n) \\ &= \prod_{n=1}^N p_\theta(\mathsf{z}_n) p_\theta(\mathsf{x}_n \mid \mathsf{z}_n). \end{aligned}$$
3. Taking logarithms gives the complete-data log-likelihood:
$$\ell_c(\theta) = \sum_{n=1}^N \left[\log p_\theta(\mathsf{z}_n) + \log p_\theta(\mathsf{x}_n \mid \mathsf{z}_n)\right].$$
4. The log-likelihood separates across observations and model components, often simplifying parameter estimation.

Why complete-data estimation is easier

1. Suppose the latent values Z were known. Parameter estimation would maximize

$$\theta_{\text{complete}} = \arg\max_{\theta} \sum_{n=1}^{N} \log p_\theta(x_n, z_n).$$

2. Using the generative factorization, each term separates into a prior contribution and a conditional observation contribution:

$$\log p_\theta(x_n, z_n) = \log p_\theta(z_n) + \log p_\theta(x_n \mid z_n).$$

3. Knowing z_n tells us which latent explanation generated each observation. This often produces simpler sufficient statistics, decomposable objectives, or closed-form updates.
4. In practice z_n is hidden. We cannot maximize the complete-data objective by treating unknown assignments as observed facts.

Marginal likelihood for observed data

1. Because z_n is unobserved, its contribution must be summed or integrated out:

$$\boxed{p_\theta(x_n) = \int p_\theta(x_n, z_n)\, dz_n.}$$

2. Under independent observations and independent local latents,

$$\boxed{\log p_\theta(\mathcal{D}) = \sum_{n=1}^{N} \log \int p_\theta(x_n, z_n)\, dz_n.}$$

3. This is the observed-data objective for maximum likelihood. The complete-data objective is not available because the latent values are unknown.
4. Notice the order of operations: first marginalize each latent variable, then take the logarithm, and finally sum over observations.

Why parameter estimation is hard with latent variables

1. The maximum likelihood objective uses only observed data:

$$\boxed{\ell(\theta) = \sum_{n=1}^{N} \log \int p_\theta(\mathrm{x}_n, \mathrm{z}_n)\, d\mathrm{z}_n.}$$

   For discrete latents, replace the integral with a sum.
2. The main difficulty is the *logarithm outside the marginalization*. Unlike complete-data learning, contributions from different latent explanations are coupled.
3. Even when $p_\theta(\mathrm{x}, \mathrm{z})$ is easy to evaluate, its integral can be high-dimensional or intractable; computing the posterior requires the same normalizing constant:

$$p_\theta(\mathrm{z} \mid \mathrm{x}) = \frac{p_\theta(\mathrm{x}, \mathrm{z})}{\int p_\theta(\mathrm{x}, \mathrm{z}')\, d\mathrm{z}'}.$$

4. Parameter estimation therefore involves two interacting tasks: *infer plausible hidden explanations* and *update the model parameters*. EM and VI organize or approximate these tasks.

Complete-data versus marginal likelihood

1. With known latent variables, the log of the joint distribution decomposes:
$$\log p_\theta(\mathcal{D}, \mathrm{Z}) = \sum_{n=1}^{N} \log p_\theta(\mathrm{x}_n, \mathrm{z}_n).$$
2. With unknown latent variables, the correct likelihood is
$$\log p_\theta(\mathcal{D}) = \sum_{n=1}^{N} \log \left[\int p_\theta(\mathrm{x}_n, \mathrm{z}_n)\, d\mathrm{z}_n \right].$$
3. The difference is conceptual: the joint likelihood evaluates one proposed latent explanation, while the marginal likelihood aggregates *all* explanations.
4. For a discrete mixture with weights π_k and component densities $p_k(\mathrm{x})$,
$$\log p_\theta(\mathrm{x}) = \log \sum_{k=1}^{K} \pi_k p_k(\mathrm{x}).$$
No single component can generally replace this sum.

The log-sum bottleneck

1. With hidden z_n, observed-data maximum likelihood requires

$$\boxed{\ell(\theta) = \sum_{n=1}^{N} \log \int p_\theta(x_n, z)\, dz.}$$

2. The integral must be evaluated *inside* the logarithm for every example. Even when $\log p_\theta(x, z)$ is easy to compute, its integral need not be.
3. For discrete latent assignments $z \in \{1, \ldots, K\}$,

$$\log p_\theta(x) = \log \sum_{k=1}^{K} p_\theta(x, z = k).$$

   The logarithm couples all latent explanations: optimizing one component changes the relative contribution of the others.
4. In high-dimensional latent spaces, exact sums can be exponentially large and continuous integrals can be analytically intractable.

Why logarithms cannot be moved inside marginalization

1. In general, the logarithm of an integral is not the integral of a logarithm:

$$\boxed{\log \int p_\theta(x, z)\, dz \neq \int \log p_\theta(x, z)\, dz.}$$

   The right-hand side is not even a posterior expectation unless a probability density is supplied as a weighting measure.

2. Intuitively, log-sum does not equal to sum-log.

EM as a bridge to variational inference

1. Expectation–maximization (EM) alternates between inference and parameter learning. At iteration k, the E-step uses the current model to compute
$$q^{(k)}(z) = p_{\theta^{(k)}}(z \mid x).$$
2. The M-step improves the expected complete-data log-likelihood:
$$\boxed{\theta^{(k+1)} \in \arg\max_{\theta} \mathbb{E}_{q^{(k)}}[\log p_\theta(x, z)].}$$
3. This separates the roles of the two unknowns:
   - E-step: assign posterior weights to latent explanations.
   - M-step: learn parameters as if those explanations were weighted observations.
4. Exact EM requires tractable posterior inference. Variational methods replace the exact posterior with a tractable approximation, preserving the same conceptual division of labor.

Review: Gaussian mixture models

1. A Gaussian mixture model represents a distribution as a weighted sum of K Gaussian components:
$$p_\theta(\mathrm{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathrm{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k).$$
2. The mixture weights satisfy
$$\pi_k \geq 0, \qquad \sum_{k=1}^{K} \pi_k = 1.$$
3. Introduce a discrete latent variable $z \in \{1, \ldots, K\}$ indicating the component:
$$p_\theta(z = k) = \pi_k,$$
$$p_\theta(\mathrm{x} \mid z = k) = \mathcal{N}(\mathrm{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k).$$
4. The complete-data likelihood is
$$p_\theta(\mathrm{x}, z = k) = \pi_k \mathcal{N}(\mathrm{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k).$$
5. The observed-data likelihood follows by marginalizing over the latent component.

Known versus unknown mixture assignments

1. Consider a Gaussian mixture model with latent component labels $z_n \in \{1, \ldots, K\}$:

$$p_\theta(\mathrm{x}_n, z_n = k) = \pi_k \mathcal{N}(\mathrm{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k).$$

2. If the component labels are known, the complete-data log-likelihood is

$$\ell_c(\theta) = \sum_{n=1}^{N} \left[\log \pi_{z_n} + \log \mathcal{N}(\mathrm{x}_n \mid \boldsymbol{\mu}_{z_n}, \boldsymbol{\Sigma}_{z_n})\right].$$

   Parameter estimation separates across components.

3. If the labels are hidden, the observed-data log-likelihood becomes

$$\ell(\theta) = \sum_{n=1}^{N} \log \left[\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathrm{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right].$$

4. The sum inside the logarithm couples the components, making direct optimization more difficult.

Marginal likelihood and posterior responsibilities

1. For independent observations $\mathcal{D} = \{x_1, \ldots, x_N\}$, the observed-data log-likelihood is
$$\log p_\theta(\mathcal{D}) = \sum_{n=1}^{N} \log \left[\sum_{k=1}^{K} \pi_k \mathcal{N}(x_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right].$$
2. The posterior probability that component k generated observation x_n is called its responsibility:
$$\gamma_{nk} = p_\theta(z_n = k \mid x_n).$$
3. By Bayes' rule,
$$\gamma_{nk} = \frac{\pi_k \mathcal{N}(x_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_n \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}.$$
4. The responsibilities satisfy
$$0 \le \gamma_{nk} \le 1, \qquad \sum_{k=1}^{K} \gamma_{nk} = 1.$$
5. They provide soft assignments of observations to mixture components.

A numerical example of responsibilities

1. Consider a one-dimensional mixture of two equally weighted Gaussians:

$$\pi_1 = \pi_2 = 0.5, \qquad \mu_1 = 0, \quad \mu_2 = 2, \quad \sigma_1^2 = \sigma_2^2 = 1.$$

2. For an observation $x = 0.5$, the two unnormalized posterior probabilities are

$$a_1 = 0.5\,\mathcal{N}(0.5 \mid 0, 1),$$
$$a_2 = 0.5\,\mathcal{N}(0.5 \mid 2, 1).$$

3. The ratio is

$$\frac{a_1}{a_2} = \exp\left(\frac{(0.5-2)^2 - (0.5-0)^2}{2}\right) = e.$$

4. Therefore,

$$\gamma_1 = \frac{e}{1+e} \approx 0.731, \qquad \gamma_2 = \frac{1}{1+e} \approx 0.269.$$

5. The observation is assigned more strongly to component 1, but both components retain nonzero responsibility.

Expectation-maximization algorithm

1. EM alternates between estimating latent variables and updating model parameters.
2. E-step: compute posterior responsibilities using the current parameters θ^{old}:
$$\gamma_{nk} = p_{\theta^{\text{old}}}(z_n = k \mid \mathrm{x}_n).$$
3. M-step: maximize the expected complete-data log-likelihood:
$$\theta^{\text{new}} = \arg\max_{\theta} Q(\theta, \theta^{\text{old}}),$$
$$Q(\theta, \theta^{\text{old}}) = \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma_{nk} \log p_{\theta}(\mathrm{x}_n, z_n = k).$$
4. For a GMM, define the effective component size:
$$N_k = \sum_{n=1}^{N} \gamma_{nk}.$$
5. The M-step has closed-form solutions for mixture weights, means, and covariances.

EM updates and interpretation

1. For a Gaussian mixture model, the M-step gives

$$\pi_k^{\text{new}} = \frac{N_k}{N}, \qquad \boldsymbol{\mu}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{nk} \mathrm{x}_n.$$

2. The covariance update is

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{nk} (\mathrm{x}_n - \boldsymbol{\mu}_k^{\text{new}}) (\mathrm{x}_n - \boldsymbol{\mu}_k^{\text{new}})^\top.$$

3. EM alternates between:
   - Inference: estimate the latent component assignments.
   - Learning: update the model parameters using those assignments.
4. Exact EM does not decrease the observed-data log-likelihood, although it may converge to a local optimum or stationary point.
5. Variational inference generalizes this idea when the exact posterior is intractable.

Outline

1 Introduction

2 Flow Matching

3 Diffusion Models

4 Score Matching

5 Connections

6 Frontiers

7 Latent Variable Models

8 Variational Inference
- Variational Inference and the ELBO
- ELBO Optimization and Posterior Approximation
- Maximizing ELBO

From latent variables to variational inference

1. A general latent-variable model is easy to specify through $p_\theta(\mathrm{x}, \mathrm{z})$, but the marginal likelihood $p_\theta(\mathrm{x})$ and posterior $p_\theta(\mathrm{z} \mid \mathrm{x})$ may be difficult to evaluate.
2. Maximum likelihood requires the evidence; learning gradients require posterior expectations:

$$\nabla_\theta \log p_\theta(\mathrm{x}) = \mathbb{E}_{p_\theta(\mathrm{z}|\mathrm{x})}[\nabla_\theta \log p_\theta(\mathrm{x}, \mathrm{z})].$$

3. Variational inference addresses both difficulties using a tractable distribution $q_\phi(\mathrm{z} \mid \mathrm{x})$ and a computable lower bound on $\log p_\theta(\mathrm{x})$.
4. We will derive the bound, interpret its gap, and explain what happens when optimizing ϕ versus θ.

Variational inference

1. Introduce an approximation $q_\phi(\mathrm{z} \mid \mathrm{x})$ to the intractable posterior $p_\theta(\mathrm{z} \mid \mathrm{x})$.
2. The model parameters θ specify the generative joint distribution; the variational parameters ϕ specify an inference distribution conditioned on observations.
3. A common approximation is mean-field factorization:

$$q_\phi(\mathrm{z} \mid \mathrm{x}) = \prod_{j=1}^{k} q_\phi(z_j \mid \mathrm{x}).$$

This simplifies inference but may miss posterior correlations.

Variational inference (continued)

1. Practical VI needs expectations and gradients of the variational objective to be tractable or estimable. Sampling from q_ϕ and evaluating its log-density are common sufficient ingredients for Monte Carlo estimators.
2. In amortized inference, a shared neural network maps each observation x to parameters of $q_\phi(\mathrm{z} \mid \mathrm{x})$, avoiding separate optimization for every data point.

Training with variational inference

1. The desired objective is the observed-data log-likelihood:

$$\max_{\theta} \sum_{n=1}^{N} \log p_\theta(\mathrm{x}_n), \qquad p_\theta(\mathrm{x}) = \int p_\theta(\mathrm{x}, \mathrm{z})\, d\mathrm{z}.$$

2. VI replaces each intractable log evidence with an evidence lower bound (ELBO) $\mathcal{L}_{\theta,\phi}(\mathrm{x})$:

$$\boxed{\mathcal{L}_{\theta,\phi}(\mathrm{x}) \leq \log p_\theta(\mathrm{x}).}$$

3. The practical training objective is

$$\boxed{\max_{\theta,\phi} \sum_{n=1}^{N} \mathcal{L}_{\theta,\phi}(\mathrm{x}_n).}$$

4. Updating ϕ improves inference for a fixed model; updating θ changes the generative model. Joint optimization couples these two goals.

Review: Jensen's inequality

1. For a convex function f, Jensen's inequality states

$$f(\mathbb{E}[X]) \leq \mathbb{E}[f(X)].$$

   The inequality reverses for concave functions.

2. Since $\log x$ is concave, its tangent line at $a > 0$ provides an upper bound:

$$\log x \leq \log a + \frac{x - a}{a}.$$

3. Choose $a = \mathbb{E}[X]$ and take expectations:

$$\begin{aligned}\mathbb{E}[\log X] &\leq \log \mathbb{E}[X] + \frac{\mathbb{E}[X] - \mathbb{E}[X]}{\mathbb{E}[X]} \\ &= \log \mathbb{E}[X].\end{aligned}$$

4. Therefore, for $X > 0$,

$$\boxed{\log \mathbb{E}[X] \geq \mathbb{E}[\log X]}.$$

5. This inequality is the key step in deriving the ELBO.

The key idea of variational inference

1. In latent variable models, we specify the joint distribution $p_\theta(x, z)$, but learning requires the marginal likelihood:

$$p_\theta(x) = \int p_\theta(x, z)\, dz.$$

2. The integral is often intractable, making direct maximization of $\log p_\theta(x)$ difficult.
3. **The variational trick:** Introduce a normalized, tractable distribution $q_\phi(z \mid x)$ and rewrite $p_\theta(x)$ in terms of $q_\phi(z \mid x)$ and $p_\theta(x, z)$ as

$$\begin{aligned} p_\theta(x) &= \int q_\phi(z \mid x) \frac{p_\theta(x, z)}{q_\phi(z \mid x)}\, dz \\ &= \mathbb{E}_{q_\phi(z|x)} \left[\frac{p_\theta(x, z)}{q_\phi(z \mid x)} \right]. \end{aligned}$$

4. $p_\theta(x, z)$ is normally tractable, and we chose a tractable $q_\phi(z \mid x)$
5. This identity allows us to apply Jensen's inequality to obtain a lower bound on the log marginal likelihood.

Deriving the evidence lower bound

1. Taking the logarithm of the identity from the previous slide:

$$\log p_\theta(\mathsf{x}) = \log \mathbb{E}_{q_\phi}\left[\frac{p_\theta(\mathsf{x},\mathsf{z})}{q_\phi(\mathsf{z}\mid\mathsf{x})}\right].$$

2. Since log is concave, Jensen's inequality gives

$$\begin{aligned}\log p_\theta(\mathsf{x}) &\geq \mathbb{E}_{q_\phi}\left[\log \frac{p_\theta(\mathsf{x},\mathsf{z})}{q_\phi(\mathsf{z}\mid\mathsf{x})}\right] \\ &= \mathbb{E}_{q_\phi}\left[\log p_\theta(\mathsf{x},\mathsf{z}) - \log q_\phi(\mathsf{z}\mid\mathsf{x})\right].\end{aligned}$$

3. Define the **evidence lower bound (ELBO)**:

$$\mathcal{L}_{\theta,\phi}(\mathsf{x}) = \mathbb{E}_{q_\phi}\left[\log p_\theta(\mathsf{x},\mathsf{z}) - \log q_\phi(\mathsf{z}\mid\mathsf{x})\right].$$

4. Therefore,

$$\boxed{\log p_\theta(\mathsf{x}) \geq \mathcal{L}_{\theta,\phi}(\mathsf{x})}.$$

Why maximizing the ELBO works

1. Bayes' rule gives

$$\log p_\theta(z \mid x) = \log p_\theta(x, z) - \log p_\theta(x).$$

2. Using the definition of KL divergence,

$$\begin{aligned} & D_{\mathrm{KL}}\left(q_\phi(z \mid x) \,\|\, p_\theta(z \mid x)\right) \\ =& \mathbb{E}_{q_\phi}\left[\log q_\phi(z \mid x) - \log p_\theta(x, z)\right] + \log p_\theta(x). \end{aligned}$$

3. Rearranging gives the exact identity:

$$\boxed{\log p_\theta(x) = \mathcal{L}_{\theta,\phi}(x) + D_{\mathrm{KL}}\left(q_\phi(z \mid x) \,\|\, p_\theta(z \mid x)\right)}.$$

4. Since KL divergence is nonnegative, the ELBO is a lower bound. It becomes exact when

$$q_\phi(z \mid x) = p_\theta(z \mid x).$$

5. Variational inference maximizes the ELBO while approximating the generally intractable posterior.

Review: Kullback–Leibler divergence

1. The KL divergence measures the discrepancy between two probability distributions:

$$D_{\mathrm{KL}}(q\|p) = \mathbb{E}_{q(\mathsf{z})}\left[\log \frac{q(\mathsf{z})}{p(\mathsf{z})}\right] = -\mathbb{E}_{q(\mathsf{z})}\left[\log \frac{p(\mathsf{z})}{q(\mathsf{z})}\right].$$

2. Its main properties are:
   - Nonnegative: $D_{\mathrm{KL}}(q\|p) \geq 0$.
   - Zero if and only if $q = p$ almost everywhere.
   - Asymmetric: generally $D_{\mathrm{KL}}(q\|p) \neq D_{\mathrm{KL}}(p\|q)$.
3. In variational inference, we compare the approximate posterior $q_\phi(\mathsf{z} \mid \mathsf{x})$ with the exact posterior $p_\theta(\mathsf{z} \mid \mathsf{x})$.

Fitting the variational posterior

1. Recall the evidence lower bound:

$$\mathcal{L}_{\theta,\phi}(\mathsf{x}) = \mathbb{E}_{q_\phi(\mathsf{z}|\mathsf{x})}\left[\log p_\theta(\mathsf{x},\mathsf{z}) - \log q_\phi(\mathsf{z}\mid\mathsf{x})\right].$$

2. Bayes' rule gives $p_\theta(\mathsf{x},\mathsf{z}) = p_\theta(\mathsf{x})p_\theta(\mathsf{z}\mid\mathsf{x})$. Substituting,

$$\begin{aligned}\mathcal{L}_{\theta,\phi}(\mathsf{x}) &= \log p_\theta(\mathsf{x}) + \mathbb{E}_{q_\phi}\left[\log \frac{p_\theta(\mathsf{z}\mid\mathsf{x})}{q_\phi(\mathsf{z}\mid\mathsf{x})}\right] \\ &= \log p_\theta(\mathsf{x}) - D_{\mathrm{KL}}\left(q_\phi(\mathsf{z}\mid\mathsf{x})\|p_\theta(\mathsf{z}\mid\mathsf{x})\right).\end{aligned}$$

3. For fixed θ, maximizing the ELBO with respect to ϕ minimizes the KL divergence to the exact posterior.

The variational gap

1. Rearranging the previous identity gives

$$\log p_\theta(\mathrm{x}) - \mathcal{L}_{\theta,\phi}(\mathrm{x}) = D_{\mathrm{KL}}(q_\phi \| p_\theta(\cdot \mid \mathrm{x})) \geq 0.$$

2. The KL term is called the **variational gap**.
3. The exact posterior and marginal likelihood may be intractable, yet the ELBO can still be evaluated or estimated from samples of q_ϕ and the joint density $p_\theta(\mathrm{x}, \mathrm{z})$.
4. If the variational family contains the exact posterior, the gap can be zero at $q_\phi(\mathrm{z} \mid \mathrm{x}) = p_\theta(\mathrm{z} \mid \mathrm{x})$.

Updating the generative model

1. Variational learning optimizes both the model parameters θ and variational parameters ϕ:

$$\max_{\theta,\phi} \sum_{n=1}^{N} \mathcal{L}_{\theta,\phi}(\mathrm{x}_n).$$

2. For each observation,

$$\mathcal{L}_{\theta,\phi}(\mathrm{x}) = \log p_\theta(\mathrm{x}) - D_{\mathrm{KL}}(q_\phi \| p_\theta(\cdot \mid \mathrm{x})).$$

3. Increasing the ELBO by changing θ can reflect:
   - An increase in the data log-likelihood.
   - A decrease in the variational gap.
   - A combination of both effects.
4. Thus, maximizing the ELBO is not necessarily identical to maximizing the data likelihood when the variational family is restricted.

Why the variational approximation affects learning

1. Updating ϕ with fixed θ changes the approximation $q_\phi(z \mid x)$, not the generative model.
2. Updating θ also changes the exact posterior $p_\theta(z \mid x)$.
3. A restricted variational family may favor model parameters whose posteriors are easier to approximate, even if other parameters achieve a higher marginal likelihood.
4. This creates a trade-off:
   - Fit the observed data well.
   - Keep posterior inference accurate within the chosen variational family.
5. A more expressive variational family can reduce this approximation-induced bias, although optimization may become harder.

Variational pruning and posterior collapse

1. When inference is restricted, learning may make some latent dimensions irrelevant to the observations.
2. For an unused latent dimension z_j, the posterior equals its prior (under an independent prior):

$$p_\theta(z_j \mid x) = p_\theta(z_j).$$

3. This behavior is called **variational pruning**; in variational autoencoders, related behavior is often called **posterior collapse**.
4. It can have both benefits and drawbacks:
   - Benefit: automatically avoids unnecessary latent dimensions.
   - Drawback: useful latent information may be discarded, limiting the model's effective capacity.
5. Posterior collapse can also arise from other causes, including an overly expressive decoder.

Choosing the variational posterior family

1. A common approximation is a factorized posterior:

$$q_\phi(\mathsf{z} \mid \mathsf{x}) = \prod_{j=1}^{d} q_\phi(z_j \mid \mathsf{x}).$$

   This is called a **mean-field approximation**.
2. More expressive alternatives include:
   - Gaussian distributions with full covariance matrices.
   - Mixture distributions.
   - Autoregressive distributions.
   - Normalizing-flow transformations of simpler distributions.
3. Greater expressiveness can reduce the variational gap, but may increase computational cost and optimization difficulty.

Amortized variational inference

1. Each observation x_n generally has a different posterior $p_\theta(z \mid x_n)$.
2. Classical variational inference may optimize separate variational parameters for every observation.
3. **Amortized inference** instead uses an inference network with shared parameters ϕ:

$$q_\phi(z \mid x), \qquad x \longmapsto \text{parameters of } q_\phi(\cdot \mid x).$$

4. The network predicts an approximate posterior for a new observation in a single forward pass.
5. The inference network and generative model are typically trained jointly by maximizing the dataset ELBO.

Approximation and amortization gaps

1. A restricted variational family may not contain the exact posterior: this creates an **approximation gap**.
2. An inference network may not find the best approximation available within that family for every observation: this creates an **amortization gap**.
3. Let $\mathcal{Q}$ be the variational family and define

$$\mathcal{L}_\theta^*(\mathrm{x}) = \sup_{q \in \mathcal{Q}} \mathcal{L}_\theta(\mathrm{x}; q).$$

   Then the total gap decomposes as

$$\begin{aligned}\log p_\theta(\mathrm{x}) - \mathcal{L}_{\theta,\phi}(\mathrm{x}) = &[\log p_\theta(\mathrm{x}) - \mathcal{L}_\theta^*(\mathrm{x})] \\ &+ [\mathcal{L}_\theta^*(\mathrm{x}) - \mathcal{L}_{\theta,\phi}(\mathrm{x})].\end{aligned}$$

4. Richer posterior families address the first gap; better inference networks or refinement can address the second.

Variational versus exact inference

1. Variational inference enables learning and posterior approximation when exact inference is intractable.
2. Main advantages:
   - Optimization-based inference using a tractable surrogate objective.
   - Fast inference for new observations with amortization.
   - Flexible latent-variable model design.
3. Main limitations:
   - Posterior approximation can be biased by the chosen family.
   - A loose ELBO may distort learning of the generative model.
   - Optimization and amortization can introduce additional errors.
4. The practical goal is to balance model fit, posterior expressiveness, inference speed, and computational cost.

Maximizing the evidence lower bound

1. Variational learning maximizes the ELBO with respect to model parameters θ and variational parameters ϕ:

$$\mathcal{L}_{\theta,\phi}(\mathsf{x}) = \mathbb{E}_{q_\phi(\mathsf{z}|\mathsf{x})} \left[\log p_\theta(\mathsf{x}, \mathsf{z}) - \log q_\phi(\mathsf{z} \mid \mathsf{x})\right].$$

2. We need two gradients:

$$\nabla_\theta \mathcal{L}_{\theta,\phi}(\mathsf{x}), \qquad \nabla_\phi \mathcal{L}_{\theta,\phi}(\mathsf{x}).$$

3. Exact expectations are often unavailable, so modern variational inference uses Monte Carlo estimates from $q_\phi(\mathsf{z} \mid \mathsf{x})$.
4. The two gradients require different treatment because the sampling distribution depends on ϕ, but not on θ.

Gradient with respect to model parameters

1. For fixed ϕ, the variational distribution does not depend on θ.
2. Differentiating the ELBO gives

$$\begin{aligned}\nabla_\theta \mathcal{L}_{\theta,\phi}(\mathrm{x}) &= \nabla_\theta \mathbb{E}_{q_\phi}[\log p_\theta(\mathrm{x},\mathrm{z}) - \log q_\phi(\mathrm{z} \mid \mathrm{x})] \\ &= \mathbb{E}_{q_\phi}[\nabla_\theta \log p_\theta(\mathrm{x},\mathrm{z})].\end{aligned}$$

3. Draw independent samples $\mathrm{z}^{(k)} \sim q_\phi(\mathrm{z} \mid \mathrm{x})$ and estimate

$$\nabla_\theta \mathcal{L}_{\theta,\phi}(\mathrm{x}) \approx \frac{1}{K} \sum_{k=1}^{K} \nabla_\theta \log p_\theta(\mathrm{x},\mathrm{z}^{(k)}).$$

4. Thus, model-parameter gradients require only samples from the variational posterior and derivatives of the log-joint model.

Why variational gradients are harder

1. When differentiating with respect to ϕ, both the integrand and the sampling distribution depend on ϕ:

$$\mathcal{L}_{\theta,\phi}(\mathrm{x}) = \int q_\phi(\mathrm{z} \mid \mathrm{x}) \left[\log p_\theta(\mathrm{x}, \mathrm{z}) - \log q_\phi(\mathrm{z} \mid \mathrm{x})\right] d\mathrm{z}.$$

2. Consequently, we cannot simply treat samples from q_ϕ as constants and differentiate the log-joint.
3. Two main strategies estimate gradients through parameter-dependent expectations:
   - Likelihood-ratio (REINFORCE) estimator: differentiate the log probability of sampled states.
   - Pathwise (reparameterization) estimator: express sampled states as differentiable transformations of fixed noise.
4. Both target the same ELBO gradient but have different applicability and variance.

Likelihood-ratio gradient estimator

1. For a function $f(z)$ independent of ϕ, use

$$\nabla_\phi q_\phi(z \mid x) = q_\phi(z \mid x) \nabla_\phi \log q_\phi(z \mid x).$$

2. Therefore,

$$\begin{aligned} \nabla_\phi \mathbb{E}_{q_\phi}[f(z)] &= \int f(z) \nabla_\phi q_\phi(z \mid x) dz \\ &= \mathbb{E}_{q_\phi}[f(z) \nabla_\phi \log q_\phi(z \mid x)]. \end{aligned}$$

3. Applied to the ELBO, the explicit derivative of $-\log q_\phi$ has zero expectation, yielding

$$\nabla_\phi \mathcal{L}_{\theta,\phi}(x) = \mathbb{E}_{q_\phi}[(\log p_\theta(x, z) - \log q_\phi(z \mid x)) \nabla_\phi \log q_\phi(z \mid x)] .$$

4. This estimator handles discrete latent variables, but can have high variance; baselines are often used to reduce it.

Reparameterization trick

1. Instead of sampling z directly from a parameter-dependent distribution, draw fixed noise:

$$\epsilon \sim p(\epsilon), \qquad \mathsf{z} = g_\phi(\epsilon, \mathsf{x}),$$

   where $p(\epsilon)$ does not depend on ϕ.
2. For a differentiable function f,

$$\mathbb{E}_{q_\phi(\mathsf{z}|\mathsf{x})}[f(\mathsf{z})] = \mathbb{E}_{p(\epsilon)}[f(g_\phi(\epsilon, \mathsf{x}))].$$

3. Differentiating through the transformation gives

$$\begin{aligned}\nabla_\phi \mathbb{E}_{q_\phi}[f(\mathsf{z})] &= \mathbb{E}_{p(\epsilon)}[\nabla_\phi f(g_\phi(\epsilon, \mathsf{x}))] \\ &= \mathbb{E}_{p(\epsilon)}\left[J_{g_\phi}^{\mathsf{T}} \nabla_{\mathsf{z}} f(\mathsf{z})\right],\end{aligned}$$

   with $\mathsf{z} = g_\phi(\epsilon, \mathsf{x})$ and $J_{g_\phi} = \partial g_\phi / \partial \phi$.
4. The parameter dependence has moved from the sampling distribution into a differentiable computation.

Pathwise gradient of the ELBO

1. Apply reparameterization to the full ELBO integrand, which itself depends on ϕ:

$$\mathcal{L}_{\theta,\phi}(\mathrm{x}) = \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\log p_\theta(\mathrm{x}, g_\phi(\boldsymbol{\epsilon}, \mathrm{x})) - \log q_\phi(g_\phi(\boldsymbol{\epsilon}, \mathrm{x}) \mid \mathrm{x})\right].$$

2. Because $p(\boldsymbol{\epsilon})$ is fixed, we may differentiate inside the expectation:

$$\nabla_\phi \mathcal{L}_{\theta,\phi}(\mathrm{x}) = \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\nabla_\phi \left(\log p_\theta(\mathrm{x}, g_\phi) - \log q_\phi(g_\phi \mid \mathrm{x})\right)\right].$$

3. Monte Carlo samples of $\boldsymbol{\epsilon}$ provide a gradient estimate that can be computed by automatic differentiation.
4. Unlike the previous slide's generic f, the ELBO integrand requires differentiating both the sample transformation and the explicit ϕ dependence of $\log q_\phi$.

Example: Gaussian reparameterization

1. Let the variational posterior be a diagonal Gaussian:

$$q_\phi(\mathsf{z} \mid \mathsf{x}) = \mathcal{N}\left(\mathsf{z} \mid \boldsymbol{\mu}_\phi(\mathsf{x}), \mathrm{diag}(\boldsymbol{\sigma}^2_\phi(\mathsf{x}))\right).$$

2. Sample standard Gaussian noise and transform it:

$$\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathsf{I}), \qquad \mathsf{z} = \boldsymbol{\mu}_\phi(\mathsf{x}) + \boldsymbol{\sigma}_\phi(\mathsf{x}) \odot \boldsymbol{\epsilon}.$$

3. The resulting z has the required Gaussian distribution, but the randomness is isolated in $\boldsymbol{\epsilon}$.
4. For each fixed $\boldsymbol{\epsilon}$, gradients propagate through $\boldsymbol{\mu}_\phi(\mathsf{x})$ and $\boldsymbol{\sigma}_\phi(\mathsf{x})$.
5. This is the standard pathwise estimator used to train variational autoencoders.

Comparing gradient estimators

1. Likelihood-ratio gradients use $\nabla_\phi \log q_\phi$ and do not require differentiating the sampled latent variable.
2. Pathwise gradients differentiate through a noise-to-sample transformation g_ϕ.

Property	**Likelihood-ratio**	**Pathwise**
Latent states	Discrete or continuous	Typically continuous
Differentiable sample map	Not needed	Needed
Differentiable integrand	Not required	Required
Typical variance	Higher	Lower

3. Reparameterization is often preferable when available; likelihood-ratio estimators remain useful for discrete variables and nondifferentiable objectives.
4. These are typical properties, not universal guarantees about estimator variance.

Which distributions can be reparameterized?

1. Location-scale distributions often admit an explicit transformation:

$$z = \mu_\phi(x) + \sigma_\phi(x) \odot \epsilon, \qquad \epsilon \sim p_0.$$

2. Examples include Gaussian, Laplace, Cauchy, and Student's t distributions with suitable base noise.
3. Other continuous distributions, including Gamma and Dirichlet families, may use implicit reparameterization methods.
4. Exact categorical samples do not generally admit a differentiable pathwise map with respect to their probabilities.
   - Likelihood-ratio estimators remain applicable.
   - Continuous relaxations can provide approximate pathwise gradients.
5. The practical choice balances applicability, gradient variance, computational cost, and approximation bias.

Variational inference: key takeaways

1. **The challenge:** Latent variable models specify the joint distribution $p_\theta(\mathrm{x}, \mathrm{z})$, but computing the marginal likelihood and exact posterior is often intractable.
2. **The solution:** Variational inference introduces a tractable approximation $q_\phi(\mathrm{z} \mid \mathrm{x})$ and maximizes the evidence lower bound (ELBO):

$$\log p_\theta(\mathrm{x}) = \mathcal{L}_{\theta,\phi}(\mathrm{x}) + D_{\mathrm{KL}}\left(q_\phi(\mathrm{z} \mid \mathrm{x}) \,\|\, p_\theta(\mathrm{z} \mid \mathrm{x})\right).$$

3. **Benefits and limitations:** VI enables scalable approximate inference and flexible model design, but introduces approximation error and depends on the variational family and optimization.
4. **Connection to earlier models:** Flow matching, diffusion, and score matching also use tractable auxiliary or conditional objectives to address difficult generative modeling problems.
5. **Unifying principle:** Construct tractable learning objectives that enable training and sampling without directly solving every intractable probabilistic computation.

Acknowledgments

The author gratefully acknowledges

1. Andrii Kryvenko for creating and maintaining the website;
2. Qilong Wang for graphics support;
3. all students for their comments;
4. supports by NSF under Grants IIS-2551110, IIS-2243850, CNS-2328395, MOMS-2331036; NIH under Grant U01AG070112; and ARPA-H under grant 1AY1AX000053.
5. supports by Texas A&M University Division of Research, and the Texas A&M Institute of Data Science.

References

1. Christopher Bishop: Deep Learning: Foundations and Concepts
2. Andriy Mnih: Modern Latent Variable Models and Variational Inference
3. MIT Course: Introduction to Flow Matching and Diffusion Models
4. Ricky Chen, Yaron Lipman, Heli Ben-Hamu: Flow Matching for Generative Modeling

THANK YOU!

Feedback?

https://divelab.github.io/GMFF/