

Report on EDG Paper

Siqi Yao

SCHOOL OF DATA SCIENCE

May 22, 2025

Table of Contents

1 Preliminaries

- Reverse-Time SDE
- Variational Inference

2 EDG Paper

- Model Overview
- Objective and Loss Function
- GHD Based Decoder

3 Thoughts

4 References

Table of Contents

1 Preliminaries

- Reverse-Time SDE
- Variational Inference

2 EDG Paper

- Model Overview
- Objective and Loss Function
- GHD Based Decoder

3 Thoughts

4 References

Forward and Reverse-Time ODE

Forward-time ODE

To simulate

$$dX_t = f(X_t, t)dt, \quad X(0) \text{ given}$$

for $t > 0$, set $X_0 = X(0)$ and compute

$$X_{k+1} = X_k + \Delta t \cdot f(X_k, k\Delta t), \quad k = 0, 1, \dots$$

for sufficiently small Δt and set $t = k\Delta t$ (**Euler-Maruyama discretization**).

Reverse-time ODE

To simulate

$$dX_t = f(X_t, t)dt, \quad X(T) \text{ given}$$

for $0 < t < T$, set $K = \lfloor T/\Delta t \rfloor$ and $X_K = X(T)$ and compute

$$X_{k-1} = X_k - \Delta t \cdot f(X_k, k\Delta t), \quad k = K, K-1, \dots, 1$$

for sufficiently small Δt and set $t = k\Delta t$.

Forward and Reverse-Time SDE?

Forward-time SDE

To simulate

$$dX_t = \mu(X_t, t)dt + \sigma(X_t, t)dW_t, \quad X_0 \sim p_0$$

for $t > 0$, sample $X_0 \sim p_0$ and compute

$$X_{k+1} = X_k + \Delta t \cdot \mu(X_k, k\Delta t) + \sigma(X_k, k\Delta t)\sqrt{\Delta t}Z_k, \quad k = 0, 1, \dots$$

for sufficiently small Δt and set $t = k\Delta t$, where $Z_1, Z_2, \dots \sim \mathcal{N}(0, I)$.

Forward and Reverse-Time SDE?

Forward-time SDE

To simulate

$$dX_t = \mu(X_t, t)dt + \sigma(X_t, t)dW_t, \quad X_0 \sim p_0$$

for $t > 0$, sample $X_0 \sim p_0$ and compute

$$X_{k+1} = X_k + \Delta t \cdot \mu(X_k, k\Delta t) + \sigma(X_k, k\Delta t)\sqrt{\Delta t}Z_k, \quad k = 0, 1, \dots$$

for sufficiently small Δt and set $t = k\Delta t$, where $Z_1, Z_2, \dots \sim \mathcal{N}(0, I)$.

Reverse-time SDE?

To simulate

$$dX_t = \mu(X_t, t)dt + \sigma(X_t, t)dW_t, \quad X_T \sim p_T$$

for $0 < t < T$, set $X_{\lfloor T/\Delta t \rfloor} = X_T$, and compute

$$X_{k-1} = X_k - \Delta t \cdot \mu(X_k, k\Delta t) - \sigma(X_k, k\Delta t)\sqrt{\Delta t}Z_k, \quad k = K, K-1, \dots, 2, 1$$

This naive method does not work! Why?

Reverse-Time SDE

To simulate reverse-time SDE, we need the theorem given by [Anderson, 1982].

Theorem 1 (Anderson's Theorem)

Given the forward-time SDE

$$dX_t = \mu(X_t, t) dt + \sigma(t) dW_t, \quad X_0 \sim p_0$$

the corresponding reverse-time SDE is

$$d\bar{X}_t = (\mu(\bar{X}_t, t) - \sigma^2(t) \nabla_x \log p_t(\bar{X}_t)) dt + \sigma(t) d\bar{W}_t, \quad \bar{X}_T \sim p_T$$

The theorem ensures that $X_t \stackrel{\mathcal{D}}{=} \bar{X}_t$ for all $0 \leq t \leq T$, meaning that the **marginal** distributions are equal for all t .

Note: If $\sigma(t)$ is a matrix, then $\sigma(t)^2$ denotes the product $\sigma(t)\sigma(t)^\top$.

Sample from Reverse-Time SDE

How to sample from the reverse-time SDE?

$$d\bar{X}_t = (\mu(\bar{X}_t, t) - \sigma^2(t) \nabla_x \log p_t(\bar{X}_t)) dt + \sigma(t) d\bar{W}_t, \quad \bar{X}_T \sim p_T$$

Sample from Reverse-Time SDE

How to sample from the reverse-time SDE?

$$d\bar{X}_t = (\mu(\bar{X}_t, t) - \sigma^2(t)\nabla_x \log p_t(\bar{X}_t)) dt + \sigma(t) d\bar{W}_t, \quad \bar{X}_T \sim p_T$$

For $0 < t < T$, sample $\bar{X}_T \sim p_T$, set $K = \lfloor T/\Delta t \rfloor$, $\bar{X}_K = \bar{X}_T$, and compute:

$$\begin{aligned} \bar{X}_{k-1} = & \bar{X}_k - \Delta t \left(\mu(\bar{X}_k, k\Delta t) \right. \\ & \left. - \sigma^2(k\Delta t)\nabla_x \log p_{k\Delta t}(\bar{X}_k) \right) + \sigma(k\Delta t)\sqrt{\Delta t}Z_k, \\ & \text{for } k = K, K-1, \dots, 2, 1 \end{aligned}$$

where $Z_1, \dots, Z_K \sim \mathcal{N}(0, I)$.

Sample from Reverse-Time SDE

How to sample from the reverse-time SDE?

$$d\bar{X}_t = (\mu(\bar{X}_t, t) - \sigma^2(t) \nabla_x \log p_t(\bar{X}_t)) dt + \sigma(t) d\bar{W}_t, \quad \bar{X}_T \sim p_T$$

For $0 < t < T$, sample $\bar{X}_T \sim p_T$, set $K = \lfloor T/\Delta t \rfloor$, $\bar{X}_K = \bar{X}_T$, and compute:

$$\begin{aligned} \bar{X}_{k-1} = & \bar{X}_k - \Delta t \left(\mu(\bar{X}_k, k\Delta t) \right. \\ & \left. - \sigma^2(k\Delta t) \nabla_x \log p_{k\Delta t}(\bar{X}_k) \right) + \sigma(k\Delta t) \sqrt{\Delta t} Z_k, \\ & \text{for } k = K, K-1, \dots, 2, 1 \end{aligned}$$

where $Z_1, \dots, Z_K \sim \mathcal{N}(0, I)$.

Alternatively, define $\{Y_t\}_{t=0}^T$ via

$$\begin{aligned} dY_t = & - \left(\mu(Y_t, T-t) - \sigma^2(T-t) \nabla_y \log p_{T-t}(Y_t) \right) dt \\ & + \sigma(T-t) dW_t, \quad Y_0 \sim p_T. \end{aligned}$$

If we set $\bar{X}_{T-t} = Y_t$, then $X_t \stackrel{\mathcal{D}}{=} \bar{X}_t = Y_{T-t}$.

Fokker-Planck Equation

Theorem 2 (Fokker-Planck Equation)

Let $X_t \in \mathbb{R}^d$ satisfy the following SDE:

$$dX_t = \mu(X_t, t) dt + \sigma(X_t, t) dW_t$$

where $\mu(x, t) \in \mathbb{R}^d$, $\sigma(x, t) \in \mathbb{R}^{d \times m}$, $W_t \in \mathbb{R}^m$

Then the probability density $p(x, t)$ of X_t satisfies the following Fokker-Planck equation:

$$\frac{\partial p(x, t)}{\partial t} = - \sum_{i=1}^d \frac{\partial}{\partial x_i} [\mu_i(x, t) p(x, t)] + \frac{1}{2} \sum_{i=1}^d \sum_{j=1}^d \frac{\partial^2}{\partial x_i \partial x_j} [D_{ij}(x, t) p(x, t)]$$

where the diffusion tensor is defined as:

$$D(x, t) = \sigma(x, t) \sigma(x, t)^\top \in \mathbb{R}^{d \times d}$$

Compact matrix form:

$$\frac{\partial p}{\partial t} = -\nabla \cdot (\mu p) + \frac{1}{2} \nabla \cdot (\nabla \cdot (Dp))$$

Proof of Anderson's Theorem: Setups

Often we can assume that the diffusion term $\sigma(t)$ does not involve X_t . However, we will prove a **more general form**. This proof is adapted from [Kim, 2025]. Consider a forward diffusion process $X_t \in \mathbb{R}^d$ on the time interval $[0, T]$, given by the SDE:

$$dX_t = f(X_t, t) dt + g(X_t, t) dW_t, \quad X_0 \text{ given},$$

where:

- ❶ W_t is a standard d -dimensional Brownian motion.
- ❷ $f : \mathbb{R}^d \times [0, T] \rightarrow \mathbb{R}^d$ is the drift coefficient.
- ❸ $g : \mathbb{R}^d \times [0, T] \rightarrow \mathbb{R}^{d \times d}$ is the diffusion matrix. We assume $g(x, t)g(x, t)^\top$ is positive definite.
- ❹ Suitable conditions (e.g., global Lipschitz) are assumed on f and g .
- ❺ $p(x, t)$, the probability density of X_t , is assumed to be sufficiently smooth and strictly positive for $t > 0$.

Then can write out the FP equation:

$$\frac{\partial p(x, t)}{\partial t} = -\nabla \cdot (f(x, t) p(x, t)) + \frac{1}{2} \Delta (g(x, t)^2 p(x, t)),$$

where Δ corresponds to applying ∇ twice.

Introduce Reverse Time

Define the reverse-time variable: $s = T - t$, i.e., $t = T - s$.

Consider the reverse-time process:

$$Y_s, \quad s \in [0, T]. \quad Y_0 \sim p_T.$$

Define:

$$F(y, s) := f(x, T - s), \quad G(y, s) := g(x, T - s), \quad q(y, s) := p(x, T - s).$$

Note that $q(y, s)$ is the probability density of Y_s . Setting $q(y, s) = p(x, T - s)$ ensures the same marginal distribution.

Remark: Although the definitions above are written using x , **we treat x and y as representing the same spatial point**. The change from x to y is purely notational and reflects the switch from the forward-time to the reverse-time representation.

Next, we rewrite the forward FP equation in terms of (y, s) using the chain rule:

$$\frac{\partial p}{\partial t}(x, t) = -\frac{\partial q}{\partial s}(y, s) = -\nabla \cdot (F(y, s) q(y, s)) + \frac{1}{2} \Delta (G(y, s)^2 q(y, s)).$$

So we obtain the reverse-time evolution equation:

$$\frac{\partial q}{\partial s}(y, s) = \nabla \cdot (F(y, s) q(y, s)) - \frac{1}{2} \Delta (G(y, s)^2 q(y, s)).$$

Postulating an SDE for the Reverse Process

According to results from stochastic calculus and probability theory, Y_s also satisfies an SDE, but driven by a different Brownian motion \widetilde{W}_s :

$$dY_s = b(Y_s, s) ds + G(Y_s, s) d\widetilde{W}_s.$$

Here, $b(x, s)$ is the unknown reverse drift coefficient that we will find out, and $G(y, s)$ is the same as what we have just defined.

The FP equation associated with this postulated reverse SDE is:

$$\frac{\partial q}{\partial s}(y, s) = -\nabla \cdot (b(y, s) q(y, s)) + \frac{1}{2} \Delta (G(y, s)^2 q(y, s)).$$

Matching Density Evolutions

For the postulated SDE to be consistent with the time-reversed forward process, the density $q(y, s)$ must satisfy *both* derived evolution equations:

$$-\nabla \cdot (b(y, s) q(y, s)) + \frac{1}{2} \Delta (G(y, s)^2 q(y, s)) = \nabla \cdot (F(y, s) q(y, s)) - \frac{1}{2} \Delta (G(y, s)^2 q(y, s)) .$$

Combining terms:

$$\nabla \cdot ([b(y, s) + F(y, s)] q(y, s)) = \Delta (G(y, s)^2 q(y, s)) .$$

Since $\Delta = \nabla \cdot \nabla$, we write:

$$\nabla \cdot ([b(y, s) + F(y, s)] q(y, s)) = \nabla \cdot (\nabla (G(y, s)^2 q(y, s))) .$$

Integrate against a **smooth test function** $\phi \in C_c^\infty(\mathbb{R}^d)$:

$$\int \nabla \cdot ([b + F] q) \phi \, dy = \int \nabla \cdot (\nabla (G^2 q)) \phi \, dy$$

By integration by parts (boundary terms vanish due to compact support of ϕ):

$$- \int ([b + F] q) \cdot \nabla \phi \, dy = - \int \nabla (G^2 q) \cdot \nabla \phi \, dy$$

Since this must hold for all ϕ , we equate the two:

$$[b(y, s) + F(y, s)] q(y, s) = \nabla (G(y, s)^2 q(y, s)) .$$

Matching Density Evolutions (cont.)

Now use the **product rule**:

$$\nabla (G(y, s)^2 q(y, s)) = [\nabla G(y, s)^2] q(y, s) + G(y, s)^2 \nabla q(y, s),$$

Substitute back:

$$[b(y, s) + F(y, s)] q(y, s) = [\nabla G(y, s)^2] q(y, s) + G(y, s)^2 \nabla q(y, s).$$

We are solving for $b(y, s)$. Rearranging terms:

$$b(y, s) q(y, s) = -F(y, s) q(y, s) + [\nabla G(y, s)^2] q(y, s) + G(y, s)^2 \nabla q(y, s)$$

Since $q(y, s) = p(y, T - s)$ is strictly positive, divide both sides by $q(y, s)$:

$$b(y, s) = -F(y, s) + \nabla G(y, s)^2 + G(y, s)^2 \frac{\nabla q(y, s)}{q(y, s)}.$$

Recognizing the last term as the score function of q :

$$\frac{\nabla q(y, s)}{q(y, s)} = \nabla \log q(y, s),$$

we obtain the final expression for the **reverse drift**:

$$b(y, s) = -F(y, s) + \nabla G(y, s)^2 + G(y, s)^2 \nabla \log q(y, s).$$

Special Case: If g depends only on time (i.e., $g(y, t) = g(t)$), then so does $G(y, s) = g(T - s)$, and $\nabla G(y, s)^2 = 0$.

Final Form of Reverse-Time SDE

Substituting the derived drift $b(y, s)$ back into the reverse SDE, we get the general reverse-time SDE for Y_s :

$$\begin{aligned} dY_s = & \left[-F(Y_s, s) + \nabla G(Y_s, s)^2 + G(Y_s, s)^2 \nabla \log q(Y_s, s) \right] ds + G(Y_s, s) d\widetilde{W}_s \\ & \stackrel{t=T-s}{=} \left[-f(Y_s, T-s) + \nabla_y (g(Y_s, T-s)^2) + g(Y_s, T-s)^2 \nabla_y \log p(Y_s, T-s) \right] ds \\ & + g(Y_s, T-s) d\widetilde{W}_s. \quad \square \end{aligned}$$

This aligns with the form on page 8.

Table of Contents

1 Preliminaries

- Reverse-Time SDE
- Variational Inference

2 EDG Paper

- Model Overview
- Objective and Loss Function
- GHD Based Decoder

3 Thoughts

4 References

Basic Ideas of Variational Inference

- Computing complex posteriors $p(z \mid x)$ is one of the cores of modern Bayesian statistics.
- Variational inference (VI) is a widely used method and has many applications in physics.

Basic Ideas of Variational Inference

- Computing complex posteriors $p(z \mid x)$ is one of the cores of modern Bayesian statistics.
- Variational inference (VI) is a widely used method and has many applications in physics.
- First define a family of densities \mathcal{Q} over latent variables z .
- Then, we find the member of \mathcal{Q} that minimizes the KL divergence to the true posterior:

$$q^*(z) = \arg \min_{q(z) \in \mathcal{Q}} D_{KL}(q(z) \parallel p(z \mid x))$$

- Finally, we approximate the posterior with the optimized distribution $q^*(z)$.

Example: VAE

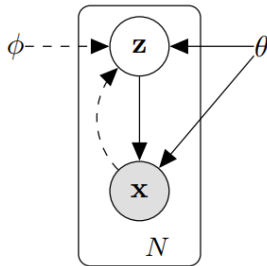


Figure 1: Graphical Model of VAE. Source: [Kingma et al., 2013]

Example: VAE

- Objective:

$$\theta^* = \arg \max_{\theta} \sum_{i=1}^n \log p_{\theta}(\mathbf{x}_i)$$

Example: VAE

- Objective:

$$\theta^* = \arg \max_{\theta} \sum_{i=1}^n \log p_{\theta}(\mathbf{x}_i)$$

- Marginalize likelihood?

$$p_{\theta}(\mathbf{x}) = \int p_{\theta}(\mathbf{x}, \mathbf{z}) d\mathbf{z} = \int p_{\theta}(\mathbf{x} | \mathbf{z}) p_{\theta}(\mathbf{z}) d\mathbf{z}$$

- Bayes's theorem?

$$p_{\theta}(\mathbf{x}) = \frac{p_{\theta}(\mathbf{x} | \mathbf{z}) p_{\theta}(\mathbf{z})}{p_{\theta}(\mathbf{z} | \mathbf{x})}$$

- Solution: Train variational approximation $q_{\phi}(\mathbf{z} | \mathbf{x})$ (also called recognition model):

$$q_{\phi}(\mathbf{z} | \mathbf{x}) \approx p_{\theta}(\mathbf{z} | \mathbf{x})$$

- VAE maximizes the ELBO:

$$\log p_{\theta}(\mathbf{x}) = \text{ELBO} + D_{\text{KL}}(q_{\phi}(\mathbf{z} | \mathbf{x}) \| p_{\theta}(\mathbf{z} | \mathbf{x}))$$

VAE and Variational Free Energy

- Energy function $q(x) = \frac{1}{Z} e^{-\beta E(x)}$. Only E is known.
- We are interested in getting samples x and the partition function Z .
- Take the model distribution $p(x)$ as a **variational distribution** and minimize the following:

$$\text{KL} \left(p(x) \left\| \frac{e^{-\beta E(x)}}{Z} \right. \right)$$

- This is equivalent to the variational free energy:

$$F = \mathbb{E}_p[E(x)] - T\mathbb{H}[p] = \int dx p(x) \left[\frac{1}{\beta} \log p(x) + E(x) \right] \geq -\frac{1}{\beta} \log Z$$

This is also known as the Gibbs-Bogoliubov-Feynman inequality.

We can rewrite the KL objectives in both cases as:

$$\text{KL} \left(q(\mathbf{z} \mid \mathbf{x}) \parallel \frac{p(\mathbf{x}, \mathbf{z})}{p(\mathbf{x})} \right) \equiv \mathbb{E}_q [\log q(\mathbf{z} \mid \mathbf{x})] - \mathbb{E}_q [\log p(\mathbf{x}, \mathbf{z})]$$

$$\begin{aligned} \text{KL} \left(p(\mathbf{x}) \parallel \frac{e^{-\beta E(\mathbf{x})}}{Z} \right) &\equiv \mathbb{E}_p [\log p(\mathbf{x})] - \mathbb{E}_p [-\beta E(\mathbf{x})] \\ &= \mathbb{E}_p [\log p(\mathbf{x})] - \mathbb{E}_p \left[\log \left(e^{-\beta E(\mathbf{x})} \right) \right] \end{aligned}$$

VAE and Variational Free Energy

Concept	VAE	VFE
Integration variable	\mathbf{z}	\mathbf{x}
Normalization constant	$p(\mathbf{x})$	Z
Integrand	$p(\mathbf{x}, \mathbf{z})$	$e^{-\beta E(\mathbf{x})}$
Objective function	$\text{KL} \left(q(\mathbf{z} \mid \mathbf{x}) \parallel \frac{p(\mathbf{x}, \mathbf{z})}{p(\mathbf{x})} \right)$	$\text{KL} \left(p(\mathbf{x}) \parallel \frac{e^{-\beta E(\mathbf{x})}}{Z} \right)$

Table 1: Correspondence between VAE and Variational Free Energy

Table of Contents

1 Preliminaries

- Reverse-Time SDE
- Variational Inference

2 EDG Paper

- Model Overview
- Objective and Loss Function
- GHD Based Decoder

3 Thoughts

4 References

Model Architecture

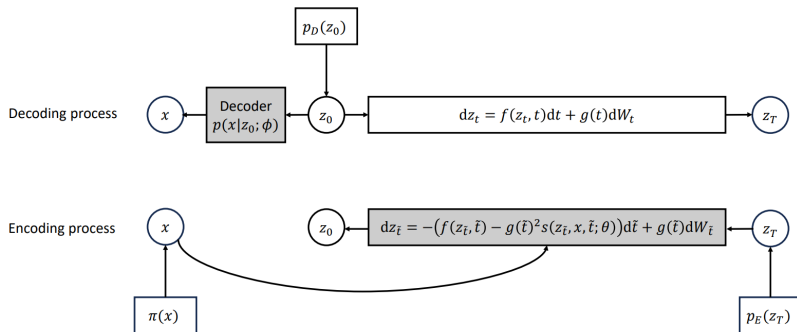


Figure 2: Model Architecture. Source: [Wang et al., 2024]

Main Steps

- **Decoder (Latent to Sample):** Sample latent prior $z_0 \in \mathbb{R}^D \sim p_D(z_0)$ and generate x

$$x \mid z_0 \sim p_D(x \mid z_0; \phi)$$

Latent z_0 also evolves via forward-time SDE:

$$dz_t = f(z_t, t) dt + g(t) dW_t, \quad t \in [0, T]$$

where W_t is the standard BM, $f(\cdot, t) : \mathbb{R}^D \rightarrow \mathbb{R}^D$, $g(\cdot) : \mathbb{R} \rightarrow \mathbb{R}$.

Main Steps

- **Decoder (Latent to Sample):** Sample latent prior $z_0 \in \mathbb{R}^D \sim p_D(z_0)$ and generate x

$$x \mid z_0 \sim p_D(x \mid z_0; \phi)$$

Latent z_0 also evolves via forward-time SDE:

$$dz_t = f(z_t, t) dt + g(t) dW_t, \quad t \in [0, T]$$

where W_t is the standard BM, $f(\cdot, t) : \mathbb{R}^D \rightarrow \mathbb{R}^D$, $g(\cdot) : \mathbb{R} \rightarrow \mathbb{R}$.

- **Encoder (Sample to Latent):** Simulate reverse path

$$dz_{\tilde{t}} = - \left(f(z_{\tilde{t}}, \tilde{t}) - g(\tilde{t})^2 \nabla_{z_{\tilde{t}}} \log p_D(z_{\tilde{t}} \mid x) \right) d\tilde{t} + g(\tilde{t}) d\bar{W}_{\tilde{t}}$$

Main Steps

- **Decoder (Latent to Sample):** Sample latent prior $z_0 \in \mathbb{R}^D \sim p_D(z_0)$ and generate x

$$x \mid z_0 \sim p_D(x \mid z_0; \phi)$$

Latent z_0 also evolves via forward-time SDE:

$$dz_t = f(z_t, t) dt + g(t) dW_t, \quad t \in [0, T]$$

where W_t is the standard BM, $f(\cdot, t) : \mathbb{R}^D \rightarrow \mathbb{R}^D$, $g(\cdot) : \mathbb{R} \rightarrow \mathbb{R}$.

- **Encoder (Sample to Latent):** Simulate reverse path

$$dz_{\tilde{t}} = - \left(f(z_{\tilde{t}}, \tilde{t}) - g(\tilde{t})^2 \nabla_{z_{\tilde{t}}} \log p_D(z_{\tilde{t}} \mid x) \right) d\tilde{t} + g(\tilde{t}) d\bar{W}_{\tilde{t}}$$

- **Score Approximation:**

$$x \sim p_D(x), \quad z_T \sim p_E(z_T) \triangleq p_D(z_T)$$

$$dz_{\tilde{t}} = - \left(f(z_{\tilde{t}}, \tilde{t}) - g(\tilde{t})^2 s(z_{\tilde{t}}, x, \tilde{t}; \theta) \right) d\tilde{t} + g(\tilde{t}) dW_{\tilde{t}}, \quad \tilde{t} = T - t$$

Main Steps

- **Decoder (Latent to Sample):** Sample latent prior $z_0 \in \mathbb{R}^D \sim p_D(z_0)$ and generate x

$$x \mid z_0 \sim p_D(x \mid z_0; \phi)$$

Latent z_0 also evolves via forward-time SDE:

$$dz_t = f(z_t, t) dt + g(t) dW_t, \quad t \in [0, T]$$

where W_t is the standard BM, $f(\cdot, t) : \mathbb{R}^D \rightarrow \mathbb{R}^D$, $g(\cdot) : \mathbb{R} \rightarrow \mathbb{R}$.

- **Encoder (Sample to Latent):** Simulate reverse path

$$dz_{\tilde{t}} = - \left(f(z_{\tilde{t}}, \tilde{t}) - g(\tilde{t})^2 \nabla_{z_{\tilde{t}}} \log p_D(z_{\tilde{t}} \mid x) \right) d\tilde{t} + g(\tilde{t}) d\bar{W}_{\tilde{t}}$$

- **Score Approximation:**

$$x \sim p_D(x), \quad z_T \sim p_E(z_T) \triangleq p_D(z_T)$$

$$dz_{\tilde{t}} = - \left(f(z_{\tilde{t}}, \tilde{t}) - g(\tilde{t})^2 s(z_{\tilde{t}}, x, \tilde{t}; \theta) \right) d\tilde{t} + g(\tilde{t}) dW_{\tilde{t}}, \quad \tilde{t} = T - t$$

- **Training Objective:**

$$D_{\text{KL}}(p_D(x, z_{[\cdot]}) \parallel p_E(x, z_{[\cdot]})) = \mathcal{L}(\theta, \phi) + \log Z$$

Table of Contents

- 1 Preliminaries
 - Reverse-Time SDE
 - Variational Inference
- 2 EDG Paper
 - Model Overview
 - Objective and Loss Function
 - GHD Based Decoder
- 3 Thoughts
- 4 References

- Goal: sample from the Boltzmann distribution

$$\pi(x) = \frac{1}{Z} \exp(-U(x)),$$

where $U : \mathbb{R}^d \rightarrow \mathbb{R}$ is energy function and the normalizing constant $Z = \int \exp(-U(x)) dx$ is intractable.

- Hope to achieve this by constructing a generator (decoder) as

$$x \mid z_0 \sim p_D(x \mid z_0; \phi),$$

$z_0 \sim p_D(z_0)$ is a tractable latent variable, and ϕ denotes the parameter of the decoder network.

- Formulate $p_D(x \mid z_0; \phi)$ as $\mathcal{N}(x \mid \mu(z_0; \phi), \Sigma(z_0; \phi))$, where both μ and Σ are represented by decoder, similar to VAE.

Theorem 3 (Data Processing Inequality)

For joint distributions $P(X, Y)$ and $Q(X, Y)$, the KL divergence satisfies:

$$D_{\text{KL}}(P(X) \parallel Q(X)) \leq D_{\text{KL}}(P(X, Y) \parallel Q(X, Y))$$

Equality holds if and only if the conditional distributions match:

$$P(Y \mid X) = Q(Y \mid X) \quad \text{for all values of } X.$$

Prove the following identity:

$$D_{\text{KL}}(P(X, Y) \parallel Q(X, Y)) = D_{\text{KL}}(P(X) \parallel Q(X)) + \mathbb{E}_{x \sim P(X)} [D_{\text{KL}}(P(Y|x) \parallel Q(Y|x))]$$

$$\begin{aligned} D_{\text{KL}}(P(X, Y) \parallel Q(X, Y)) &= \sum_{x,y} P(x, y) \log \frac{P(x, y)}{Q(x, y)} \\ &= \sum_{x,y} P(x)P(y|x) \log \left[\frac{P(x)}{Q(x)} \cdot \frac{P(y|x)}{Q(y|x)} \right] \\ &= \sum_{x,y} P(x)P(y|x) \left[\log \frac{P(x)}{Q(x)} + \log \frac{P(y|x)}{Q(y|x)} \right] \\ &= \underbrace{\sum_x P(x) \log \frac{P(x)}{Q(x)}}_{D_{\text{KL}}(P(X) \parallel Q(X))} + \sum_x P(x) \underbrace{\sum_y P(y|x) \log \frac{P(y|x)}{Q(y|x)}}_{D_{\text{KL}}(P(Y|x) \parallel Q(Y|x))} \\ &= D_{\text{KL}}(P(X) \parallel Q(X)) + \mathbb{E}_{x \sim P(X)} [D_{\text{KL}}(P(Y|x) \parallel Q(Y|x))] \quad \square \end{aligned}$$

Objective Function

Objective function:

$$D_{\text{KL}}(p_D(x) \parallel \pi(x))$$

Applying data processing inequality, we can derive an upper bound:

$$D_{\text{KL}}(p_D(x) \parallel \pi(x)) \leq D_{\text{KL}}(p_D(z_0) \cdot p_D(x \mid z_0, \phi) \parallel \pi(x) \cdot p_E(z_0 \mid x, \theta))$$

Equality holds if and only if the conditional distributions match:

$$p_D(z_0 \mid x) = p_E(z_0 \mid x)$$

Objective Function

Objective function:

$$D_{\text{KL}}(p_D(x) \parallel \pi(x))$$

Applying data processing inequality, we can derive an upper bound:

$$D_{\text{KL}}(p_D(x) \parallel \pi(x)) \leq D_{\text{KL}}(p_D(z_0) \cdot p_D(x \mid z_0, \phi) \parallel \pi(x) \cdot p_E(z_0 \mid x, \theta))$$

Equality holds if and only if the conditional distributions match:

$$p_D(z_0 \mid x) = p_E(z_0 \mid x)$$

So we minimize the KL divergence of the joint distribution instead:

$$D_{\text{KL}}(p_D(x, z_{[\cdot]}) \parallel p_E(x, z_{[\cdot]}))$$

where $z_{[\cdot]}$ denotes the path $\{z_t\}_{t \in [0, T]}$ from 0 to T .

The problem now turns to **how to construct the encoding process**.

Main Steps

- **Decoder (Latent to Sample):** Latent z_0 evolves via forward-time SDE:

$$dz_t = f(z_t, t) dt + g(t) dW_t, \quad t \in [0, T]$$

and obtain z_T

- **Encoder (Sample to Latent):** Simulate reverse path

$$dz_{\tilde{t}} = - \left(f(z_{\tilde{t}}, \tilde{t}) - g(\tilde{t})^2 \nabla_{z_{\tilde{t}}} \log p_D(z_{\tilde{t}} | x) \right) d\tilde{t} + g(\tilde{t}) d\bar{W}_{\tilde{t}}$$

starting from z_T

- **Score Approximation:**

$$x \sim p_D(x), \quad z_T \sim p_E(z_T) \triangleq p_D(z_T)$$

$$dz_{\tilde{t}} = - \left(f(z_{\tilde{t}}, \tilde{t}) - g(\tilde{t})^2 s(z_{\tilde{t}}, x, \tilde{t}; \theta) \right) d\tilde{t} + g(\tilde{t}) dW_{\tilde{t}}, \quad \tilde{t} = T - t$$

- **Training Objective:**

$$D_{\text{KL}}(p_D(x, z_{[\cdot]}) \parallel p_E(x, z_{[\cdot]})) = \mathcal{L}(\theta, \phi) + \log Z$$

Theorem 4 (Loss Function)

$$D_{\text{KL}}(p_D(x, z_{[\cdot]}) \parallel p_E(x, z_{[\cdot]})) = \mathcal{L}(\theta, \phi) + \log Z,$$

where

$$\begin{aligned} \mathcal{L}(\theta, \phi) = & \mathbb{E}_{p_D} [\log p_D(x \mid z_0; \phi) + U(x)] \\ & + \int_0^T \frac{g(t)^2}{2} \mathbb{E}_{p_D} \left[\|s(z_t, x, t; \theta)\|^2 + 2 \operatorname{tr} \left(\frac{\partial s(z_t, x, t)}{\partial z_t} \right) + \|\nabla_{z_t} \log p_D(z_t)\|^2 \right] dt. \end{aligned}$$

Loss Function Derivation

The proof is adapted from [Wang et al., 2024].

Let timestep $\tau = T/L$, where L is a large number. Then:

$$\begin{aligned} & \mathbb{E}_{p_D} \left[\log \frac{p_D(x, z_0, z_\tau, \dots, z_T)}{p_E(x, z_0, z_\tau, \dots, z_T)} \right] \\ &= \mathbb{E}_{p_D} \left[\log \frac{p_D(x | z_0) p_D(z_T) \prod_{l=1}^L p_D(z_{(l-1)\tau} | z_{l\tau})}{\pi(x) p_E(z_T) \prod_{l=1}^L p_E(z_{(l-1)\tau} | z_{l\tau}, x)} \right] \\ &= \mathbb{E}_{p_D} \left[\log \frac{p_D(x | z_0)}{Z^{-1} \exp(-U(x))} \right] + \sum_{l=1}^L \Delta_{l\tau} \\ &= \mathbb{E}_{p_D} [\log p_D(x | z_0) + U(x)] + \log Z + \sum_{l=1}^L \Delta_{l\tau} \end{aligned}$$

where

$$\Delta_t \triangleq \mathbb{E}_{p_D} [\log p_D(z_{t-\tau} | z_t)] - \mathbb{E}_{p_D} [\log p_E(z_{t-\tau} | z_t, x)]$$

Next we find $\mathbb{E}_{p_D} [\log p_D(z_{t-\tau} | z_t)]$ and $\mathbb{E}_{p_D} [\log p_E(z_{t-\tau} | z_t, x)]$.

Loss Function Derivation

For small τ , the discretization of $z_{[\cdot]}$ in the decoding process provides:

$$\begin{aligned} z_t &= z_{t-\tau} + f(z_{t-\tau}, t - \tau) \tau + \sqrt{\tau} g(t - \tau) u_{t-\tau} \\ z_{t-\tau} &= z_t - \left(f(z_t, t) - g(t)^2 \nabla \log p_D(z_t) \right) \tau + \sqrt{\tau} g(t) \bar{u}_t \end{aligned}$$

where $u_{t-\tau}, \bar{u}_t \sim \mathcal{N}(\cdot \mid 0, I)$. $u_{t-\tau}$ is independent of $\{z_{t'} \mid t' \leq t - \tau\}$ and \bar{u}_t is independent of $\{z_{t'} \mid t' \geq t\}$.

The discretization of $p_E(z_{[\cdot]} \mid x)$ gives:

$$z_{t-\tau} = z_t - \left(f(z_t, t) - g(t)^2 s(z_t, x, t) \right) \tau + \sqrt{\tau} g(t) v_t$$

with $v_t \sim \mathcal{N}(\cdot \mid 0, I)$.

Loss Function Derivation

$$\begin{aligned}\mathbb{E}_{p_D} [\log p_D(z_{t-\tau} \mid z_t)] &= -\frac{d}{2} \log 2\pi - \frac{d}{2} \log \tau g(t)^2 \\ &\quad - \frac{1}{2} \tau^{-1} g(t)^{-2} \mathbb{E}_{p_D} \left[\left\| z_{t-\tau} - z_t + (f(z_t, t) - g(t)^2 \nabla \log p_D(z_t)) \tau \right\|^2 \right] \\ &= -\frac{d}{2} \log 2\pi - \frac{d}{2} \log \tau g(t)^2 - \frac{1}{2} \tau^{-1} g(t)^{-2} \mathbb{E}_{p_D} \left[\left\| \sqrt{\tau} g(t) \bar{u}_t \right\|^2 \right] \\ &= -\frac{d}{2} \log 2\pi - \frac{d}{2} \log \tau g(t)^2 - \frac{d}{2}\end{aligned}$$

Loss Function Derivation

$$\begin{aligned}\mathbb{E}_{p_D} [\log p_E(z_{t-\tau} \mid z_t, x)] &= -\frac{d}{2} \log 2\pi - \frac{d}{2} \log \tau g(t)^2 \\ &\quad - \frac{1}{2} \tau^{-1} g(t)^{-2} \mathbb{E}_{p_D} \left[\|z_{t-\tau} - z_t + (f(z_t, t) - g(t)^2 s(z_t, x)) \tau\|^2 \right] \\ &= -\frac{d}{2} \log 2\pi - \frac{d}{2} \log \tau g(t)^2 \\ &\quad - \frac{1}{2} \tau^{-1} g(t)^{-2} \mathbb{E}_{p_D} \left[\|z_{t-\tau} - z_t + f(z_t, t) \tau\|^2 \right] \\ &\quad - \frac{1}{2} \tau^{-1} g(t)^{-2} \mathbb{E}_{p_D} \left[\|g(t)^2 s(z_t, x) \tau\|^2 \right] \\ &\quad + \tau^{-1} g(t)^{-2} \mathbb{E}_{p_D} \left[(z_{t-\tau} - z_t + f(z_t, t) \tau)^\top g(t)^2 s(z_t, x) \tau \right]\end{aligned}$$

Loss Function Derivation

$$\begin{aligned} &= -\frac{d}{2} \log 2\pi - \frac{d}{2} \log \tau g(t)^2 \\ &\quad - \frac{1}{2} \tau^{-1} g(t)^{-2} \mathbb{E}_{p_D} \left[\|g(t)^2 \nabla \log p_D(z_t) \tau + \sqrt{\tau} g(t) \bar{u}_t\|^2 \right] \\ &\quad - \frac{1}{2} \tau g(t)^2 \mathbb{E}_{p_D} \left[\|s(z_t, x)\|^2 \right] \\ &\quad + \mathbb{E}_{p_D} \left[(z_{t-\tau} - z_t + f(z_t, t) \tau)^\top s(z_t, x) \right] \\ \\ &= -\frac{d}{2} \log 2\pi - \frac{d}{2} \log \tau g(t)^2 - \frac{d}{2} \\ &\quad - \frac{1}{2} \tau g(t)^2 \mathbb{E}_{p_D} \left[\|\nabla \log p_D(z_t)\|^2 \right] \\ &\quad - \frac{1}{2} \tau g(t)^2 \mathbb{E}_{p_D} \left[\|s(z_t, x)\|^2 \right] \\ &\quad + \mathbb{E}_{p_D} \left[(z_{t-\tau} - z_t + f(z_t, t) \tau)^\top s(z_t, x) \right] \end{aligned}$$

Taylor Expansion (Part 1)

Next we find $(z_{t-\tau} - z_t + f(z_t, t)\tau)$ and $s(z_t, x)$.

$$\begin{aligned} z_{t-\tau} - z_t + f(z_t, t)\tau &= z_{t-\tau} - (z_{t-\tau} + f(z_{t-\tau}, t-\tau)\tau + \sqrt{\tau}g(t-\tau)u_{t-\tau}) \\ &\quad + f(z_t, t)\tau \\ &= -f(z_{t-\tau}, t-\tau)\tau - \sqrt{\tau}g(t-\tau)u_{t-\tau} + \tau f(z_{t-\tau}, t-\tau) \\ &\quad + \tau \frac{\partial f(z_{t-\tau}, t-\tau)}{\partial z_{t-\tau}} (z_t - z_{t-\tau}) + \tau^2 \frac{\partial f(z_{t-\tau}, t-\tau)}{\partial t} \\ &\quad + \tau \cdot \mathcal{O}(\|z_t - z_{t-\tau}\|^2 + \tau^2) \\ &= -\sqrt{\tau}g(t-\tau)u_{t-\tau} + \tau \frac{\partial f(z_{t-\tau}, t-\tau)}{\partial z_{t-\tau}} (f(z_{t-\tau}, t-\tau)\tau + \sqrt{\tau}g(t-\tau)u_{t-\tau}) \\ &\quad + \tau^2 \frac{\partial f(z_{t-\tau}, t-\tau)}{\partial t} + \tau \cdot \mathcal{O}(\|f(z_{t-\tau}, t-\tau)\tau + \sqrt{\tau}g(t-\tau)u_{t-\tau}\|^2 + \tau^2) \\ &= -\sqrt{\tau}g(t-\tau)u_{t-\tau} + \mathcal{O}(\tau^2) \end{aligned}$$

Where we used the first order Taylor expansion:

$$f(z_t, t) \approx f(z_{t-\tau}, t-\tau) + \frac{\partial f(z_{t-\tau}, t-\tau)}{\partial z_{t-\tau}} (z_t - z_{t-\tau}) + \frac{\partial f(z_{t-\tau}, t-\tau)}{\partial t} \cdot \tau + \mathcal{O}(\|z_t - z_{t-\tau}\|^2 + \tau^2)$$

Taylor Expansion (Part 2)

$$\begin{aligned} s(z_t, x, t) &= s(z_{t-\tau}, x, t - \tau) + \frac{\partial s(z_{t-\tau}, x, t - \tau)}{\partial z_{t-\tau}} (z_t - z_{t-\tau}) + \frac{\partial s(z_{t-\tau}, x, t - \tau)}{\partial t} \tau \\ &= s(z_{t-\tau}, x, t - \tau) + \frac{\partial s(z_{t-\tau}, x, t - \tau)}{\partial z_{t-\tau}} (f(z_{t-\tau}, t - \tau)\tau + \sqrt{\tau}g(t - \tau)u_{t-\tau}) \\ &\quad + \frac{\partial s(z_{t-\tau}, x, t - \tau)}{\partial t} \tau \\ &= s(z_{t-\tau}, x, t - \tau) + \sqrt{\tau} \frac{\partial s(z_{t-\tau}, x, t - \tau)}{\partial z_{t-\tau}} g(t - \tau)u_{t-\tau} + \mathcal{O}(\tau) \end{aligned}$$

Final Combination

Combining the two equations above, we get

$$\begin{aligned}\mathbb{E}_{p_D} \left[(z_{t-\tau} - z_t + f(z_t, t)\tau)^\top s(z_t, x, t) \right] &= \mathbb{E}_{p_D} \left[-\sqrt{\tau}g(t-\tau)u_{t-\tau}^\top s(z_{t-\tau}, x, t-\tau) \right] \\ &\quad - \tau g(t-\tau)^2 \mathbb{E}_{p_D} \left[u_{t-\tau}^\top \frac{\partial s(z_{t-\tau}, x, t-\tau)}{\partial z_{t-\tau}} u_{t-\tau} \right] + o(\tau) \\ &= -\tau g(t-\tau)^2 \mathbb{E}_{p_D} \left[\text{tr} \left(\frac{\partial s(z_{t-\tau}, x, t-\tau)}{\partial z_{t-\tau}} \right) \right] + o(\tau)\end{aligned}$$

Finally we obtain:

$$\begin{aligned}\Delta_t &\triangleq \mathbb{E}_{p_D} [\log p_D(z_{t-\tau} \mid z_t)] - \mathbb{E}_{p_D} [\log p_E(z_{t-\tau} \mid z_t, x)] \\ &= \frac{1}{2} \tau g(t)^2 \mathbb{E}_{p_D} \left[\|\nabla \log p_D(z_t)\|^2 \right] + \frac{1}{2} \tau g(t)^2 \mathbb{E}_{p_D} \left[\|s(z_t, x, t)\|^2 \right] \\ &\quad + \tau g(t-\tau)^2 \mathbb{E}_{p_D} \left[\text{tr} \left(\frac{\partial s(z_{t-\tau}, x, t-\tau)}{\partial z_{t-\tau}} \right) \right] + o(\tau)\end{aligned}$$

$$\begin{aligned}\mathbb{E}_{p_D} \left[\log \frac{p_D(x, z_{[\cdot]})}{p_E(x, z_{[\cdot]})} \right] &= \lim_{L \rightarrow \infty} \mathbb{E}_{p_D} \left[\log \frac{p_D(x, z_0, z_\tau, \dots, z_T)}{p_E(x, z_0, z_\tau, \dots, z_T)} \right] \\&= \mathbb{E}_{p_D} [\log p_D(x \mid z_0) + U(x)] + \log Z \\&\quad + \int \frac{g(t)^2}{2} \mathbb{E}_{p_D} \left[\|s(z_t, x, t)\|^2 \right. \\&\quad \left. + 2 \operatorname{tr} \left(\frac{\partial s(z_t, x, t)}{\partial z_t} \right) + \|\nabla \log p_D(z_t)\|^2 \right] dt \quad \square\end{aligned}$$

Hutchinson Estimator

For a vector \mathbf{z} formed by *i.i.d.* zero-mean, unit-variance random variables, we have

$$\mathbb{E}[\mathbf{z}\mathbf{z}^\top] = \mathbf{I}$$

Then we can derive the so called Hutchinson estimator for trace:

$$\begin{aligned}\text{tr}(\mathbf{A}) &= \text{tr}(\mathbf{A}\mathbf{I}) = \text{tr}(\mathbf{A}\mathbb{E}[\mathbf{z}\mathbf{z}^\top]) \\ &= \mathbb{E}[\text{tr}(\mathbf{A}\mathbf{z}\mathbf{z}^\top)] = \mathbb{E}[\mathbf{z}^\top \mathbf{A} \mathbf{z}]\end{aligned}$$

\mathbf{z} is often sampled from Rademacher distribution or standard normal distribution.

Applying Hutchinson Estimator

By utilizing the importance-sampled integration and Hutchinson estimator, we can obtain an equivalent expression of $\mathcal{L}(\theta, \phi)$ which can be estimated efficiently:

$$\begin{aligned}\mathcal{L}(\theta, \phi) &= \mathbb{E}_{p_D} [\log p_D(x \mid z_0; \phi) + U(x)] \\ &\quad + \mathbb{E}_{t \sim p(t)} [\lambda(t) \cdot \mathbb{E}_{x, z_t \sim p_D, \epsilon} [\mathcal{L}_t(x, z_t, \epsilon; \theta)]]\end{aligned}$$

$$\begin{aligned}\text{where } \mathcal{L}_t(x, z_t, \epsilon; \theta) &= \|s(z_t, x, t; \theta)\|^2 \\ &\quad + 2 \cdot \frac{\partial [\epsilon^\top s(z_t, x, t; \theta)]}{\partial z_t} \epsilon \\ &\quad + \|\nabla_{z_t} \log p_D(z_t)\|^2,\end{aligned}$$

$p(\epsilon)$ is the Rademacher distribution, $p(t)$ is a proposal distribution of $t \in [0, T]$, and the weighting function is $\lambda(t) = \frac{g(t)^2}{2p(t)}$.

$$\begin{aligned} & \int_0^T \frac{g(t)^2}{2} \mathbb{E}_{p(\epsilon) p_D(x, z_t)} [\mathcal{L}_t(x, z_t, \epsilon)] dt \\ &= \int_0^T \left(\frac{g(t)^2}{2} \cdot \frac{1}{p(t)} \right) \cdot \mathbb{E}_{p(\epsilon) p_D(x, z_t)} [\mathcal{L}_t(x, z_t, \epsilon)] \cdot p(t) dt \\ &= \mathbb{E}_{t \sim p(t)} [\lambda(t) \cdot \mathbb{E}_{p(\epsilon) p_D(x, z_t)} [\mathcal{L}_t(x, z_t, \epsilon)]] \end{aligned}$$

By the principle of importance sampling, the proposal distribution $p(t)$ should **ideally match the shape of the integrand** to reduce variance.

$$p(t) \propto g(t)^2 \cdot \mathbb{E}_{p(\epsilon) p_D(x, z_t)} [\mathcal{L}_t(x, z_t, \epsilon; \theta)]$$

Finally, by sampling over $t \sim p(t)$ and $(x, z_t) \sim p_D$, we obtain the final estimate.

Implementing Importance-Sampled Integration

We illustrate the implementation of importance-weighted time sampling with a toy example.

- Suppose $T = 10$, and we divide the time range into 5 bins:

$$[0, 2), \quad [2, 4), \quad [4, 6), \quad [6, 8), \quad [8, 10)$$

- Assume we recorded the following 5 samples from recent training:

t	\mathcal{L}_t	bin
1.2	2.5	bin 1
3.1	1.2	bin 2
1.8	3.1	bin 1
7.4	0.8	bin 4
5.9	1.6	bin 3

- For each bin, we compute the average $\mathcal{L}_t \cdot g(t)^2$. If $g(t)$ is not constant, we use the actual t value of each sample:
 - bin 1: $2.5 \cdot g(1.2)^2, 3.1 \cdot g(1.8)^2 \Rightarrow \text{average} = \frac{2.5 \cdot g(1.2)^2 + 3.1 \cdot g(1.8)^2}{2}$
 - bin 2: $1.2 \cdot g(3.1)^2$
 - bin 3: $1.6 \cdot g(5.9)^2$
 - bin 4: $0.8 \cdot g(7.4)^2$
 - bin 5: no data \Rightarrow assign small positive constant to avoid zero probability
- We then normalize the values across all 5 bins to obtain a discrete proposal distribution $p(t)$ for the next round of time sampling.

Table of Contents

1 Preliminaries

- Reverse-Time SDE
- Variational Inference

2 EDG Paper

- Model Overview
- Objective and Loss Function
- GHD Based Decoder

3 Thoughts

4 References

Markov Chain Monte Carlo (MCMC)

- Another method to sample from intractable distributions.
- Starts by taking a random draw z_0 from some initial distribution $q(z_0)$ or $q(z_0 \mid x)$.
- Apply a stochastic transition operator (also called transition kernel) to the random draw z_0 :

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

- By designing the transition operator and applying it many times, z_T will converge in distribution to the exact posterior $p(z \mid x)$.

Markov Chain Monte Carlo (MCMC)

- Another method to sample from intractable distributions.
- Starts by taking a random draw z_0 from some initial distribution $q(z_0)$ or $q(z_0 | x)$.
- Apply a stochastic transition operator (also called transition kernel) to the random draw z_0 :

$$z_t \sim q(z_t | z_{t-1}, x).$$

- By designing the transition operator and applying it many times, z_T will converge in distribution to the exact posterior $p(z | x)$.

Method	Advantages	Disadvantages
VI	Fast	Less accurate
MCMC	Accurate	Slow

Table 2: Comparison between Variational Inference and MCMC

Metropolis-Hastings Algorithm

- **Step 1: Initialization**

- Choose an initial state x_0 (randomly or based on prior knowledge).

- **Step 2: Propose a Candidate Sample**

- Draw x' from the proposal distribution $Q(x' | x_t)$.

- **Step 3: Compute Acceptance Probability**

- Compute

$$A(x', x_t) = \min \left(1, \frac{P(x')}{P(x_t)} \cdot \frac{Q(x_t | x')}{Q(x' | x_t)} \right)$$

- $P(x)$: target distribution (unnormalized is okay).
- Why does $A(x', x_t)$ take this form? **Detailed balance condition.**

- **Step 4: Accept or Reject**

- Draw $u \sim \mathcal{U}[0, 1]$.
- If $u < A(x', x_t)$, accept: $x_{t+1} = x'$.
- Else, reject: $x_{t+1} = x_t$.

- **Step 5: Iterate**

- Repeat Steps 2–4 to generate a sequence x_0, x_1, \dots, x_N .

Hamiltonian Monte Carlo (HMC) Algorithm: Motivation

- **Motivation and Advantages**

- *Problem:* Random walk behavior and low efficiency of traditional MCMC.
- HMC Advantage:
 - Leverages gradient information of the target distribution to reduce inefficient exploration.
 - Simulates physical dynamics to escape local modes.

Hamiltonian Monte Carlo (HMC) Algorithm: Motivation

• Motivation and Advantages

- *Problem:* Random walk behavior and low efficiency of traditional MCMC.
- HMC Advantage:
 - Leverages gradient information of the target distribution to reduce inefficient exploration.
 - Simulates physical dynamics to escape local modes.

• Core Intuition: Analogy to Physical Systems

- Imagine a frictionless landscape with hills and valleys:
 - **Potential Energy (U):** Defined as negative log probability, $U(q) = -\log P(q)$.
Low potential regions (valleys) \Leftrightarrow high-probability regions.
High potential regions (peaks) \Leftrightarrow low-probability regions.
 - **Kinetic Energy (K):** Energy from momentum, helps particle overcome barriers.
 - **Total Energy (Hamiltonian):** $H(q, p) = U(q) + K(p)$, where q is position, p is momentum.
- **Main Idea:** HMC preserves momentum to avoid random walk behavior.

HMC Algorithm Overview

• Step 1: Initialization

- Start at current position q_t , and randomly draw momentum $p \sim \mathcal{N}(0, M)$, where M is the mass matrix (usually symmetric/diagonal and positive semidefinite, e.g., identity matrix).

• Step 2: Simulate Hamiltonian Dynamics

- Use **leapfrog method** to simulate dynamics and generate a proposal (q', p') :

① Half-step momentum update: $p \leftarrow p - \frac{\epsilon}{2} \nabla_q U(q)$

② Full-step position update: $q \leftarrow q + \epsilon M^{-1} p$

③ Another half-step momentum update: $p \leftarrow p - \frac{\epsilon}{2} \nabla_q U(q)$

- Repeat above steps L times (trajectory length = $L \times \epsilon$)

• Step 3: Accept or Reject

- (Optional) Negate the momentum: $p' \leftarrow -p'$.
- Compute change in Hamiltonian: $\Delta H = H(q', p') - H(q_t, p)$
- Accept with probability: $A = \min(1, \exp(-\Delta H))$
- If accepted, let $q_{t+1} = q'$, otherwise keep $q_{t+1} = q_t$

• Step 4: Iterate

- Discard momentum p , retain position q as the sample
- Repeat Steps 1–3 to generate a sequence of samples

HMC Explained: Hamiltonian Dynamics

A $2d$ -dimensional state space consists of d -dimensional **position vector** q and d -dimensional **momentum vector** p . The system is governed by a Hamiltonian function $H(q, p)$. For HMC, we usually use the form:

$$H(q, p) = U(q) + K(p)$$

where $U(q)$ is the *potential energy*:

$$U(q) = -\log P(q)$$

and $K(p)$ is the *kinetic energy*, typically defined as:

$$K(p) = \frac{1}{2} p^\top M^{-1} p = \sum_{i=1}^d \frac{p_i^2}{2m_i}, \quad \text{if } M = \text{diag}(m_1, \dots, m_d)$$

HMC Explained: Hamilton's Equations

The evolution of q and p over time t follows **Hamilton's equations**:

$$\frac{dq_i}{dt} = \frac{\partial H}{\partial p_i}, \quad \frac{dp_i}{dt} = -\frac{\partial H}{\partial q_i}$$

for $i = 1, \dots, d$.

With $H(q, p) = U(q) + K(p)$, the equations become:

$$\frac{dq_i}{dt} = [M^{-1}p]_i, \quad \frac{dp_i}{dt} = -\frac{\partial U}{\partial q_i}$$

HMC Explained: the Leapfrog Method

Hamiltonian dynamics is continuous. How to discretize?

Euler's method is a simple but unstable way, for $i = 1, \dots, d$:

$$p_i(t + \varepsilon) = p_i(t) + \varepsilon \frac{dp_i}{dt}(t) = p_i(t) - \varepsilon \frac{\partial U}{\partial q_i}(q(t))$$

$$q_i(t + \varepsilon) = q_i(t) + \varepsilon \frac{dq_i}{dt}(t) = q_i(t) + \varepsilon \frac{p_i(t)}{m_i}$$

HMC Explained: the Leapfrog Method

Hamiltonian dynamics is continuous. How to discretize?

Euler's method is a simple but unstable way, for $i = 1, \dots, d$:

$$p_i(t + \varepsilon) = p_i(t) + \varepsilon \frac{dp_i}{dt}(t) = p_i(t) - \varepsilon \frac{\partial U}{\partial q_i}(q(t))$$

$$q_i(t + \varepsilon) = q_i(t) + \varepsilon \frac{dq_i}{dt}(t) = q_i(t) + \varepsilon \frac{p_i(t)}{m_i}$$

Leapfrog method is widely used (more stable, preserves key properties of Hamiltonian dynamics):

$$p_i(t + \varepsilon/2) = p_i(t) - (\varepsilon/2) \frac{\partial U}{\partial q_i}(q(t))$$

$$q_i(t + \varepsilon) = q_i(t) + \varepsilon \frac{p_i(t + \varepsilon/2)}{m_i}$$

$$p_i(t + \varepsilon) = p_i(t + \varepsilon/2) - (\varepsilon/2) \frac{\partial U}{\partial q_i}(q(t + \varepsilon))$$

HMC Explained: MCMC from Hamilton Dynamics

- In statistical mechanics, the **canonical distribution** over states has the form:

$$P(x) = \frac{1}{Z} \exp\left(-\frac{E(x)}{T}\right)$$

where $E(x)$ is energy, T is temperature, and Z is a normalizing constant.

- In HMC, the **Hamiltonian** $H(q, p)$ acts as a total energy:

$$P(q, p) = \frac{1}{Z} \exp(-H(q, p))$$

- Since $H(q, p) = U(q) + K(p)$, we get independent canonical distributions:

$$P(q, p) = \frac{1}{Z} \exp(-U(q)) \exp(-K(p))$$

HMC Explained: Acceptance Probability

Since the proposal distribution is symmetric:

$$q(z' | z) = q(z | z') \Rightarrow \frac{q(z | z')}{q(z' | z)} = 1$$

Then the Metropolis acceptance probability simplifies to:

$$\alpha(z \rightarrow z') = \min \left(1, \frac{\pi(z')}{\pi(z)} \right) = \min \left(1, \frac{\exp(-H(q^*, -p^*))}{\exp(-H(q, p))} \right)$$

Using the property that kinetic energy is even:

$$\begin{aligned} K(p) = K(-p) &\Rightarrow H(q^*, -p^*) = U(q^*) + K(p^*) \\ \Rightarrow \alpha &= \min(1, \exp(-U(q^*) - K(p^*) + U(q) + K(p))) \end{aligned}$$

• Step 1: Initialization

- Start at current position q_t , and randomly draw momentum $p \sim \mathcal{N}(0, M)$, where M is the mass matrix (usually symmetric/diagonal and positive semidefinite, e.g., identity matrix).

• Step 2: Simulate Hamiltonian Dynamics

- Use numerical integration (e.g., Leapfrog) to simulate dynamics and generate a proposal (q', p') :
 - 1 Half-step momentum update: $p \leftarrow p - \frac{\epsilon}{2} \nabla_q U(q)$
 - 2 Full-step position update: $q \leftarrow q + \epsilon M^{-1} p$
 - 3 Another half-step momentum update: $p \leftarrow p - \frac{\epsilon}{2} \nabla_q U(q)$
- Repeat above steps L times (trajectory length = $L \times \epsilon$)

• Step 3: Accept or Reject

- (Optional) Negate the momentum: $p' \leftarrow -p'$.
- Compute change in Hamiltonian: $\Delta H = H(q', p') - H(q_t, p)$
- Accept with probability: $A = \min(1, \exp(-\Delta H))$
- If accepted, let $q_{t+1} = q'$, otherwise keep $q_{t+1} = q_t$

• Step 4: Iterate

- Discard momentum p , retain position q as the sample
- Repeat Steps 1–3 to generate a sequence of samples

L2HMC (Learning to HMC)

- HMC still has disadvantages:
 - Limited expressiveness of fixed leapfrog integrator.
 - Poor mixing in multi-modal or low-density regions.

L2HMC (Learning to HMC)

- HMC still has disadvantages:
 - Limited expressiveness of fixed leapfrog integrator.
 - Poor mixing in multi-modal or low-density regions.
- **L2HMC** was proposed in [Levy et al., 2017], which augments the HMC state with:
 - Binary direction $d \in \{-1, 1\}$.
 - Binary mask $m^t \in \{0, 1\}^n$, selecting coordinates for partial updates. Drawn uniformly from binary vectors satisfying $\sum_{i=1}^n m_i^t = \lfloor \frac{n}{2} \rfloor$.
 - Define $\bar{m}^t = \mathbf{1} - m^t$ and $x_{m^t} = x \odot m^t$ (where \odot denotes element-wise multiplication and $\mathbf{1}$ is the all-ones vector).
- The augmented state is $\xi = (x, v, d)$ with joint density $p(\xi) = p(x)p(v)p(d)$, where x is the position and $v \sim \mathcal{N}(0, I)$ is the momentum. Each update uses a learned leapfrog operator \mathcal{L}_θ .
- We will focus on the **modified leapfrog updates** of L2HMC.

L2HMC: Learned Leapfrog Integrator

- Modified leapfrog updates (for single timestep t and direction $d = 1$):
- **Each sub-update only depends on part of the state:**

$$\zeta_1 \triangleq (x, \partial_x U(x), t)$$

$$\zeta_2 \triangleq (x_{\bar{m}^t}, v, t)$$

$$\zeta_3 \triangleq (x'_{m^t}, v, t)$$

$$\zeta_4 \triangleq (x'', \partial_x U(x''), t)$$

- **Momentum update (Step 1):**

$$v' = v \odot \exp\left(\frac{\varepsilon}{2} S_v(\zeta_1)\right) - \frac{\varepsilon}{2} (\partial_x U(x) \odot \exp(\varepsilon Q_v(\zeta_1)) + T_v(\zeta_1))$$

- **Position update (Step 2):**

$$x' = x_{\bar{m}^t} + m^t \odot [x \odot \exp(\varepsilon S_x(\zeta_2)) + \varepsilon(v' \odot \exp(\varepsilon Q_x(\zeta_2)) + T_x(\zeta_2))]$$

$$x'' = x'_{m^t} + \bar{m}^t \odot [x' \odot \exp(\varepsilon S_x(\zeta_3)) + \varepsilon(v' \odot \exp(\varepsilon Q_x(\zeta_3)) + T_x(\zeta_3))]$$

- **Final momentum update (Step 3):**

$$v'' = v' \odot \exp\left(\frac{\varepsilon}{2} S_v(\zeta_4)\right) - \frac{\varepsilon}{2} (\partial_x U(x'') \odot \exp(\varepsilon Q_v(\zeta_4)) + T_v(\zeta_4))$$

L2HMC: Learned Leapfrog Integrator

- In each sub-update, L2HMC applies three **learned transformations** (MLPs):
 - T : a translation applied to x or v .
 - Q : a scaling term applied to the gradient.
 - S : a scaling term applied directly to x or v .
- These learned scalings bring several benefits:
 - S can accelerate transitions in low-density regions and improve mode mixing.
 - Q allows better conditioning of the energy landscape.
 - Scaling can also partially suppress noisy or rapidly oscillating gradients.

Algorithm 1 GHD based decoder

Require: $z_0 = (\zeta, v_1, \dots, v_K)$ drawn from the standard Gaussian distribution and decoder parameters ϕ

```

1: Let  $y := a(\zeta; \phi)$ .
2: for  $k = 1, \dots, K$  do
3:   for  $j = 1, \dots, J$  do
4:     Let  $s := \frac{(k-1)J+j-1}{KJ}$ .
5:     Udata  $(y, v_i)$  by
        
$$v_k := v_k - \frac{\epsilon(s; \phi)}{2} \left( \nabla U(y) \odot e^{\frac{\epsilon_0}{2} Q_v(y, \nabla U(y), s; \phi)} + T_v(y, \nabla U(y), s; \phi) \right)$$

        
$$y := y + \epsilon(s; \phi) \left( v_k \odot e^{\epsilon_0 Q_y(v_k, s; \phi)} + T_y(v_k, s; \phi) \right)$$

        
$$v_k := v_k - \frac{\epsilon(s; \phi)}{2} \left( \nabla U(y) \odot e^{\frac{\epsilon_0}{2} Q_v(y, \nabla U(y), s; \phi)} + T_v(y, \nabla U(y), s; \phi) \right)$$

6:   end for
7:   Let  $v_k := -v_k$ .
8: end for
9: Let  $\mu = y - e^{\epsilon_0 \eta(y; \phi)} \nabla U(y)$  and  $\Sigma = 2e^{\epsilon_0 \eta(y; \phi)} I$ .
10: return  $x \sim \mathcal{N}(\mu, \Sigma)$ .
```

Figure 3: GHD Decoder. Source: [Wang et al., 2024]

Why not Directly Return Sample?

Final step of the decoder:

$$\mu = y - e^{\epsilon_0 \eta(y; \phi)} \nabla U(y), \quad \Sigma = 2e^{\epsilon_0 \eta(y; \phi)} I$$

Similar to **Langevin dynamics**!

Recall the Langevin update step:

$$x \leftarrow y - \alpha \nabla U(y) + \sqrt{2\alpha} \cdot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I)$$

Now compare with the EDG decoder's final sampling step:

Langevin step	EDG decoder form
α	$e^{\epsilon_0 \eta(y; \phi)}$
$-\alpha \nabla U(y)$	$-e^{\epsilon_0 \eta(y; \phi)} \nabla U(y)$
$\sqrt{2\alpha} \cdot \epsilon$	$2e^{\epsilon_0 \eta(y; \phi)} I$

Table 3: Correspondence between Langevin dynamics and EDG decoder

Therefore, this step is essentially an **adaptive Langevin step + sampling**, not a traditional decoder outputting samples directly.

Why not Directly Return Sample?

- **Problem with standard EDG decoder:**

- Uses deterministic mapping $y = a(z)$, which encourages mode seeking.
- Final samples y tend to collapse into high-density regions of the target distribution.

- **This paper's solution:** add a final Langevin-inspired noise smoothing step.

- Use $\nabla U(y)$ to indicate the energy gradient direction.
- Use $\eta(y)$ to modulate the noise scale (trainable scalar function).
- This avoids collapse and improves sample coverage.

Possible Improvements?

- Perform Langevin steps on x obtained from the decoder. Inspired by [Xie et al., 2021].
- Use normalizing flow model as decoder.
- Try various score matching techniques.

References I



Anderson, B. D. (1982).

Reverse-time diffusion equation models.

Stochastic Processes and their Applications, 12(3):313–326.



Ansari, A. F. (2020).

Monte carlo sampling using langevin dynamics.

<https://abdufatir.com/blog/2020/Langevin-Monte-Carlo/>.

Accessed: 2025-05-18.



Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017).

Variational inference: A review for statisticians.

Journal of the American statistical Association, 112(518):859–877.



Kim, J.-H. (2025).


Deriving reverse-time stochastic differential equations (sdes).


[https://jiha-kim.github.io/posts/](https://jiha-kim.github.io/posts/deriving-reverse-time-stochastic-differential-equations-sdes/)


[deriving-reverse-time-stochastic-differential-equations-sdes](https://jiha-kim.github.io/posts/deriving-reverse-time-stochastic-differential-equations-sdes/)

References II

 Kingma, D. P., Welling, M., et al. (2013).
Auto-encoding variational bayes.

 Levy, D., Hoffman, M. D., and Sohl-Dickstein, J. (2017).
Generalizing hamiltonian monte carlo with neural networks. arxiv
2017.
arXiv preprint arXiv:1711.09268.

 Neal, R. M. et al. (2011).
Mcmc using hamiltonian dynamics.
Handbook of markov chain monte carlo, 2(11):2.

 Ryu, E. K. (2024).
Diffusion models chapter 1: Reverse-time sde.
<https://ernestryu.com/courses/FM/diffusion1.pdf>.
Lecture notes, Generative AI and Foundation Models, Seoul
National University.



Salimans, T., Kingma, D., and Welling, M. (2015).

Markov chain monte carlo and variational inference: Bridging the gap.

In *International conference on machine learning*, pages 1218–1226. PMLR.



Wang, L. and Zhang, P. (2018).

Generative models for physicists.

Technical report, Tech. rep., Institute of Physics, Chinese Academy of Sciences, GitHub. io.



Wang, Y., Guo, L., Wu, H., and Zhou, T. (2024).

Energy based diffusion generator for efficient sampling of boltzmann distributions.

arXiv preprint arXiv:2401.02080.



Xie, J., Zheng, Z., and Li, P. (2021).

Learning energy-based model with variational auto-encoder as amortized sampler.

In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10441–10451.

Thank you!
Any questions?