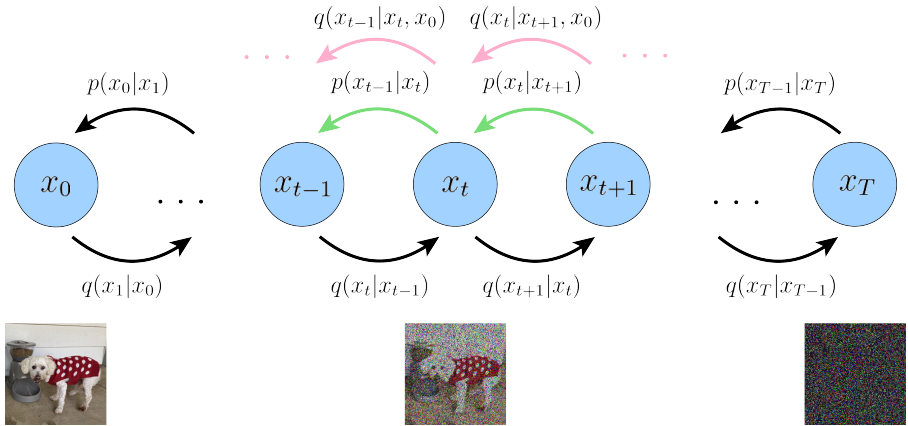
Lecture 15. Diffusion Models

Bao Wang
Department of Mathematics
Scientific Computing and Imaging Institute
University of Utah
Math 5750/6880, Fall 2023

Diffusion Models



- **Forward diffusion process:** $q(x_t|x_{t-1}), \forall t = 1, 2, \ldots, T$.
- **Reverse diffusion process:** $q(x_{t-1}|x_t, x_0), \forall t = T, T-1, \ldots, 1$.
- Diffusion model learns the reverse process $p(x_{t-1}|x_t)$ to match $q(x_{t-1}|x_t, x_0)$ for training data x_0s and $t = T, \ldots, 1$. The learned reverse process can be applied to new sampled x_T to generate new data.

Forward Diffusion Process

• Given a sample from a data distribution $\boldsymbol{x}_0 \sim q(\boldsymbol{x})$, we define a **forward diffusion process** by adding a small amount of Gaussian noise in T steps (with variance schedule $\{\beta_t \in (0,1)\}_{t=1}^{T}$):

$$q(\mathsf{x}_t|\mathsf{x}_{t-1}) = \mathcal{N}(\mathsf{x}_t; \sqrt{1-\beta_t}\mathsf{x}_{t-1}, \beta_t \mathsf{I}) \quad q(\mathsf{x}_{1:T}|\mathsf{x}_0) = \prod_{t=1}^{T} q(\mathsf{x}_t|\mathsf{x}_{t-1})$$

• Let $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$, we have

$$q(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1-\bar{\alpha}_t)\boldsymbol{I})$$

Usually, $\beta_1 < \beta_2 < \ldots < \beta_T$. In DDPM, $\beta_1 = 10^{-4}$ and $\beta_T = 0.02$. When $T \to \infty$, $\boldsymbol{x}_T$ is equivalent to an isotropic Gaussian distribution.

Reverse Diffusion Process

• If we can reverse the above process and sample from $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$, we will be able to recreate the true sample from a Gaussian noise input, $\boldsymbol{x}_T \sim \mathcal{N}(0, \mathrm{I})$.

• We cannot estimate $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ because it needs to use the entire dataset, therefore, we need to learn a model p_θ to approximate these conditional probabilities to run the **reverse diffusion process**.

$$p_\theta(\mathrm{x}_{0:T}) = p(\mathrm{x}_T)\prod_{t=1}^{T} p_\theta(\mathrm{x}_{t-1}|\mathrm{x}_t) \quad p_\theta(\mathrm{x}_{t-1}|\mathrm{x}_t) = \mathcal{N}(\mathrm{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathrm{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathrm{x}_t, t))$$

• The reverse conditional probability is tractable when conditioned on $\boldsymbol{x}_0$. Use Bayes' rule, we have

$$\begin{aligned} q(\mathrm{x}_{t-1}|\mathrm{x}_t, \mathrm{x}_0) &= q(\mathrm{x}_t|\mathrm{x}_{t-1}, \mathrm{x}_0)\frac{q(\mathrm{x}_{t-1}|\mathrm{x}_0)}{q(\mathrm{x}_t|\mathrm{x}_0)} \\ &\propto \exp\Big(-\frac{1}{2}\big(\big(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}\big)\mathrm{x}_{t-1}^2 - \big(\frac{2\sqrt{\alpha_t}}{\beta_t}\mathrm{x}_t + \frac{2\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}\mathrm{x}_0\big)\mathrm{x}_{t-1} + C(\mathrm{x}_t, \mathrm{x}_0)\big)\Big) \\ &:= \mathcal{N}(\mathrm{x}_{t-1}; \boldsymbol{\mu}_q(\mathrm{x}_t, \mathrm{x}_0), \tilde{\beta}_t \mathrm{I}) \end{aligned}$$

where $C(\boldsymbol{x}_t, \boldsymbol{x}_0)$ is some function not involving $\boldsymbol{x}_{t-1}$ and the details are omitted.

Reverse Diffusion Process

- The mean and variance can be parameterized as ($\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{T} \alpha_i$):

$$\tilde{\beta}_t = 1/(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}) = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t$$

$$\tilde{\boldsymbol{\mu}}_q(\mathrm{x}_t, \mathrm{x}_0) = (\frac{\sqrt{\alpha_t}}{\beta_t}\mathrm{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}\mathrm{x}_0)/(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}})$$

$$= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathrm{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathrm{x}_0$$

- Note that $q(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1-\bar{\alpha}_t)\boldsymbol{I})$, we can represent then $\boldsymbol{x}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}(\boldsymbol{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t)$; thus

$$\boldsymbol{\mu}_q = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathrm{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\frac{1}{\sqrt{\bar{\alpha}_t}}(\mathrm{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t) = \frac{1}{\sqrt{\alpha_t}}\left(\mathrm{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_t\right)$$

Training: Maximize the Evidence Lower Bound (ELBO)

- $$\log p_\theta(\boldsymbol{x}) = \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}\Big[\log \frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{q(\boldsymbol{z}|\boldsymbol{x})}\Big] + D_{\mathrm{KL}}(q(\boldsymbol{z}|\boldsymbol{x})||p_\theta(\boldsymbol{z}|\boldsymbol{x})) \geq \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}\Big[\log \frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{q(\boldsymbol{z}|\boldsymbol{x})}\Big]$$

- For the multi-level case, we have

$$L_{\mathrm{VLB}} := \mathbb{E}_{q(\mathrm{x}_{\mathbf{0}:T})}\Big[\log \frac{q(\mathrm{x}_{1:T}|\mathrm{x}_0)}{p_\theta(\mathrm{x}_{0:T})}\Big] \geq -\mathbb{E}_{q(\mathrm{x}_{\mathbf{0}})} \log p_\theta(\mathrm{x}_0)$$

We need to minimize L_{VLB}.

- Notice that we have

$$L_{\mathrm{VLB}} = \mathbb{E}_q\Big[\underbrace{D_{\mathrm{KL}}(q(\mathrm{x}_T|\mathrm{x}_0) \parallel p_\theta(\mathrm{x}_T))}_{L_T} + \sum_{t=2}^{T} \underbrace{D_{\mathrm{KL}}(q(\mathrm{x}_{t-1}|\mathrm{x}_t,\mathrm{x}_0) \parallel p_\theta(\mathrm{x}_{t-1}|\mathrm{x}_t))}_{L_{t-\mathbf{1}}} \underbrace{- \log p_\theta(\mathrm{x}_0|\mathrm{x}_1)}_{L_{\mathbf{0}}}\Big]$$

- Every KL term in L_{VLB} (except for L_0) compares two Gaussian distributions and can be computed in closed form. L_T is constant and can be ignored during training because q has no learnable parameters and $\boldsymbol{x}_T$ is a Gaussian noise. [May not be true for finite T]

Parameterization of L_t for Training Loss

• We need to learn a neural network to approximate the conditioned probability distributions in the reverse diffusion process, $p_\theta(\mathrm{x}_{t-1}|\mathrm{x}_t) = \mathcal{N}(\mathrm{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathrm{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathrm{x}_t, t))$.

• We train $\boldsymbol{\mu}_\theta$ to predict $\boldsymbol{\mu}_q = \frac{1}{\sqrt{\alpha_t}}\left(\mathrm{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_t\right)$. Because $\boldsymbol{x}_t$ is available as input at training time, we can reparameterize the Gaussian noise term instead to predict $\boldsymbol{\epsilon}_t$ from the input $\boldsymbol{x}_t$ at time step t:

$$\boldsymbol{\mu}_\theta(\mathrm{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(\mathrm{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathrm{x}_t, t)\right)$$

• The loss term L_t is parameterized to minimize the KL divergence (closed form of KL divergence):

$$\begin{aligned} L_t &= \mathbb{E}_{\mathbf{x_0},\boldsymbol{\epsilon}}\left[\frac{1}{2\|\boldsymbol{\Sigma}_\theta(\mathrm{x}_t, t)\|_2^2}\|\boldsymbol{\mu}_q(\mathrm{x}_t, \mathrm{x}_0) - \boldsymbol{\mu}_\theta(\mathrm{x}_t, t)\|^2\right] \\ &= \mathbb{E}_{\mathbf{x_0},\boldsymbol{\epsilon}}\left[\frac{(1-\alpha_t)^2}{2\alpha_t(1-\bar{\alpha}_t)\|\boldsymbol{\Sigma}_\theta\|_2^2}\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathrm{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t, t)\|^2\right] \end{aligned}$$

• DDPM chooses to fix β_t as constants instead of making them learnable and set $\boldsymbol{\Sigma}_\theta(\mathrm{x}_t, t) = \sigma_t^2\mathbf{I}$, where σ_t is not learned but set to β_t or $\tilde{\beta}_t = \frac{\mathbf{1}-\bar{\alpha}_{t-\mathbf{1}}}{\mathbf{1}-\bar{\alpha}_t} \cdot \beta_t$.

Training and Sampling DDPM

- Empirically, training DDPM works better with a simplified objective that ignores the weighting term:

$$L_t^{\text{simple}} = \mathbb{E}_{t\sim[1,T],\mathbf{x_0},\boldsymbol{\epsilon}_t}\left[\left\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathrm{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t, t)\right\|^2\right]$$

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\right\|^2$$
6: **until** converged

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t\mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

Latent Diffusion Model

• The latent diffusion model (LDM) runs the diffusion process in the latent space instead of pixel space, making training cost lower and inference faster.

• An encoder $\mathcal{E}$ is used to compress the input image $\mathrm{x} \in \mathbb{R}^{H \times W \times 3}$ to a smaller 2D latent vector $\mathrm{z} = \mathcal{E}(\mathrm{x}) \in \mathbb{R}^{h \times w \times c}$. Then an decoder $\mathcal{D}$ reconstructs the images from the latent vector, $\tilde{\boldsymbol{x}} = \mathcal{D}(\boldsymbol{z})$.

• The diffusion and denoising processes happen on the latent vector $\boldsymbol{z}$. The denoising model is a time-conditioned U-Net, augmented with the cross-attention mechanism to handle flexible conditioning information for image generation (e.g. class labels).

L_t: A Score Viewpoint

- Tweedie's formula: $\mathbb{E}[\boldsymbol{\mu}_q|\boldsymbol{x}_t] = \boldsymbol{x}_t + (1-\bar{\alpha}_t)\nabla \log p(\boldsymbol{x}_t)$, where $\nabla \log p(\boldsymbol{x}_t) := \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t)$.

- We have shown that $q(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1-\bar{\alpha}_t)\boldsymbol{I})$. Then, Tweedie's formula implies that

$$\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 = \boldsymbol{x}_t + (1-\bar{\alpha}_t)\nabla \log p(\boldsymbol{x}_t) \Rightarrow \boldsymbol{x}_0 = \frac{\boldsymbol{x}_t + (1-\bar{\alpha}_t)\nabla \log p(\boldsymbol{x}_t)}{\sqrt{\bar{\alpha}_t}}.$$

- The ground-truth denoising transition mean $\boldsymbol{\mu}_q(\boldsymbol{x}_t, \boldsymbol{x}_0)$ again and derive a new form:

$$\boldsymbol{\mu}_q(\boldsymbol{x}_t, \boldsymbol{x}_0) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\boldsymbol{x}_0}{1-\bar{\alpha}_t} = \frac{1}{\sqrt{\alpha_t}}\boldsymbol{x}_t + \frac{1-\alpha_t}{\sqrt{\alpha_t}}\nabla \log p(\boldsymbol{x}_t)$$

- Therefore, we can set our approximate denoising transition mean $\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t)$ as follows:

$$\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\boldsymbol{x}_t + \frac{1-\alpha_t}{\sqrt{\alpha_t}}\boldsymbol{s}_\theta(\boldsymbol{x}_t, t).$$

- Then L_t can be written as follows (score matching):

$$L_t = D_{KL}\big(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)||p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)\big) = \frac{1}{2\|\boldsymbol{\Sigma}_\theta\|_2^2}\frac{(1-\alpha_t)^2}{\alpha_t}\Big[\|\boldsymbol{s}_\theta(\boldsymbol{x}_t, t) - \nabla \log p(\boldsymbol{x}_t)\|_2^2\Big]$$

Here, $\boldsymbol{s}_\theta(\boldsymbol{x}_t, t)$ is a neural network that learns to predict the score function $\nabla \log p(\boldsymbol{x}_t)$.

Some Score Matching Algorithms

• **Explicit score matching:** learns θ so that $\psi(\boldsymbol{x};\theta)$ best matches the corresponding score of the true distribution, i.e. $\frac{\partial \log p(\boldsymbol{x})}{\partial \boldsymbol{x}}$. The corresponding objective function is

$$J_{ESM_p}(\theta) = \mathbb{E}_{p(\boldsymbol{x})}\Big[\frac{1}{2}\Big\|\psi(\boldsymbol{x};\theta) - \frac{\partial \log p(\boldsymbol{x})}{\partial \boldsymbol{x}}\Big\|^2\Big],$$

since p is unknown, we do not have explicit regression targets $\frac{\partial \log p(\boldsymbol{x})}{\partial \boldsymbol{x}}$.

• **Implicit score matching:** Notice that

$$\underbrace{\mathbb{E}_{p(\boldsymbol{x})}\Big[\frac{1}{2}\Big\|\psi(\boldsymbol{x};\theta) - \frac{\partial \log p(\boldsymbol{x})}{\partial \boldsymbol{x}}\Big\|^2\Big]}_{J_{ESM_p(\theta)}} = \underbrace{\mathbb{E}_{p(\boldsymbol{x})}\Big[\frac{1}{2}\|\psi(\boldsymbol{x};\theta)\|^2 + \sum_{i=1}^{d}\frac{\partial \psi_i(\boldsymbol{x};\theta)}{\partial \boldsymbol{x}_i}\Big]}_{J_{ISM_p(\theta)}} + C_1,$$

where $\psi_i(\boldsymbol{x};\theta) = \psi(\boldsymbol{x};\theta)_i$ and C_1 is a constant that does not depend on θ. This yields an implicit score matching objective J_{ISM_p} that no longer requires having an explicit score target for p.

- **Denoising score matching:**

  > Given i.i.d. samples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)}\} \sim p(\boldsymbol{x})$, the Parzen window density estimator is

$$p_\sigma(\tilde{\boldsymbol{x}}) = \frac{1}{n}\sum_{t=1}^{n} p_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x}^{(t)}) = \frac{1}{n}\sum_{t=1}^{n} \frac{1}{(2\pi)^{d/2}\sigma^d} e^{-\frac{1}{2\sigma^2}\|\tilde{\boldsymbol{x}}-\boldsymbol{x}^{(t)}\|^2}$$

  > Explicitly matching $\psi(\boldsymbol{x};\theta)$ with the score of Parzen windows density estimator gives

$$J_{ESM_{p_\sigma}}(\theta) = \mathbb{E}_{p_\sigma(\tilde{\boldsymbol{x}})}\left[\frac{1}{2}\left\|\psi(\tilde{\boldsymbol{x}};\theta) - \frac{\partial \log p_\sigma(\tilde{\boldsymbol{x}})}{\partial \boldsymbol{x}}\right\|^2\right].$$

  > Consider pairs of clean and corrupted data $(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \sim p_\sigma(\tilde{\boldsymbol{x}}, \boldsymbol{x}) = p_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x})p_0(\boldsymbol{x})$, we define the following denoising score matching (DSM) objective:

$$J_{DSM_{p_\sigma}}(\theta) = \mathbb{E}_{p_\sigma(\boldsymbol{x},\tilde{\boldsymbol{x}})}\left[\frac{1}{2}\left\|\psi(\tilde{\boldsymbol{x}};\theta) - \frac{\partial \log p_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x})}{\partial \tilde{\boldsymbol{x}}}\right\|^2\right].$$

  With Gaussian kernel, we have $\frac{\partial \log p_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x})}{\partial \tilde{\boldsymbol{x}}} = \frac{1}{\sigma^2}(\boldsymbol{x} - \tilde{\boldsymbol{x}})$.

- ESM and ISM directly samples from the true distribution. In contrast, DSM approximates the Parzen window density estimator from the samples – Gaussian mixture.

Yuhao Huang, Qingsong Wang, Akwum Onwunta, Bao Wang, Efficient Score Matching via Deep Equilibrium Layers, ICLR, 2024.

Score-based Generative Models

Score-based Generative Models

- We choose a sequence of noise levels $\{\sigma_t\}_{t=1}^T$ and define a sequence of perturbed data distributions:

$$p_{\sigma_t}(\boldsymbol{x}_t) = \int p(\boldsymbol{x})\mathcal{N}(\boldsymbol{x}_t; \boldsymbol{x}, \sigma_t^2 \boldsymbol{I}) d\boldsymbol{x}.$$

Notice that none of $p_{\sigma_t}(\boldsymbol{x}_t)$ is a Gaussian mixture, and DSM can be worse than SSM.

- Then, a neural network $\boldsymbol{s}_\theta(\boldsymbol{x}, t)$ is used to learn the score function for all noise levels simultaneously:

$$\arg\min_\theta \sum_{t=1}^T \lambda(t) \mathbb{E}_{p_{\sigma_t}(\boldsymbol{x}_t)} \left[\| \boldsymbol{s}_\theta(\boldsymbol{x}, t) - \nabla \log p_{\sigma_t}(\boldsymbol{x}_t) \|_2^2 \right],$$

where $\lambda(t)$ is a positive weighting function that conditions on noise level t.

- Furthermore, we use annealed Langevin dynamics sampling as a generative procedure, in which samples are produced by running Langevin dynamics for each $t = T, \ldots, 2, 1$ in sequence.

- The initialization is chosen from a fixed prior, e.g. uniform, and each subsequent sampling step starts from the final samples of the previous simulation. The noise levels steadily decrease over timesteps t, and we reduce the step size over time. [Some related theoretical analysis.]

Score SDE Formulation

- **Forward SDE:** The diffusion process can be viewed as the following SDE:

$$d\boldsymbol{x} = f(\boldsymbol{x}, t)dt + g(t)d\boldsymbol{w}, \;\; t \in [0, T],$$

where $p_0(\boldsymbol{x}) = p(\boldsymbol{x})$ is the data distribution. After perturbing $p(\boldsymbol{x})$ for a sufficiently long time T, $p_T(\boldsymbol{x})$ becomes close to a tractable noise distribution $\pi(\boldsymbol{x})$, called a **prior distribution**.

- **Reverse SDE:** The sampling is done via a corresponding reverse-time SDE of the forward process:

$$d\boldsymbol{x} = [f(\boldsymbol{x}, t) - g(t)^2 \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})]d\bar{t} + g(t)d\bar{\boldsymbol{w}}, \;\; t \in [0, T],$$

where $\bar{t}$ denotes time traveling backward from T to 0 and $\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})$ is the score function.

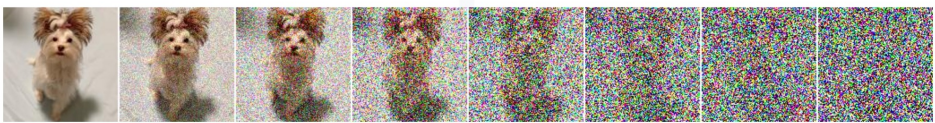
- The training objective is a weighted sum of score-matching objective:

$$L := \mathbb{E}_t \Big\{ \lambda(t) \mathbb{E}_{\boldsymbol{x_0}} \mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x_0})} \Big[\|\boldsymbol{s}_\theta(\boldsymbol{x}, t) - \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t|\boldsymbol{x}_0)\|_2^2 \Big] \Big\}$$

where $\boldsymbol{x}_0$ is sampled from the data distribution p_0 and $\lambda(t)$ is sampled from the data distribution p_0 and $\lambda(t)$ is the positive weighting function. $q(\boldsymbol{x}_t|\boldsymbol{x}_0)$ is the Gaussian transition kernel associated with the forward process. For example, $q(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t; \boldsymbol{x}_0, \sigma^2(t)\boldsymbol{I})$.

Forward SDE (data → noise)

$\mathbf{x}(0)$ —— $\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}$ ——→ $\mathbf{x}(T)$



score function

$\mathbf{x}(0)$ ←— $\mathrm{d}\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - g^2(t)\nabla_{\mathbf{x}} \log p_t(\mathbf{x})\right] \mathrm{d}t + g(t)\mathrm{d}\bar{\mathbf{w}}$ —— $\mathbf{x}(T)$

Reverse SDE (noise → data)

Probability flow ODE

• **Probability flow ODE (PFO):** PFO is the continuous-time ODE that supports the deterministic process which shares the same marginal probability density with SDE. The corresponding PFO is

$$d\boldsymbol{x} = \left[f(\boldsymbol{x}, t) - \frac{1}{2} g(t)^2 \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}) \right] dt.$$

PFO can be solved with larger step sizes as they have no randomness.

• Problem: Enforcing guidance in the SDE design for particular applications.

Reference:
`https://arxiv.org/abs/2209.15408`

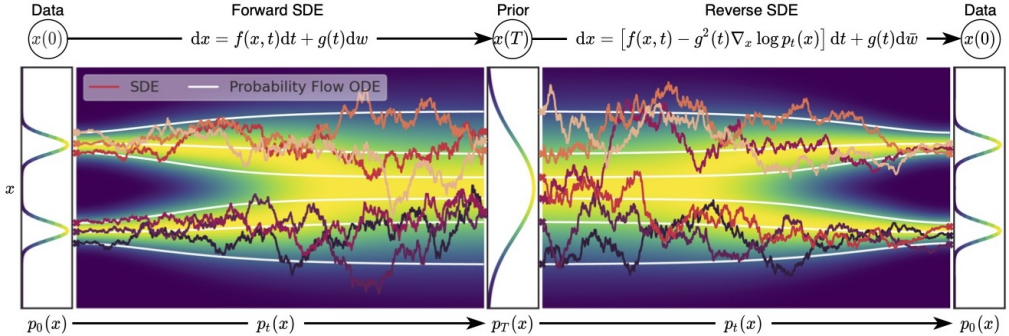


Figure: We map data to a noise distribution (prior) with an SDE, and reverse this SDE for generative modeling. We can also reverse the associated probability flow ODE, yielding a deterministic process that samples from the same distribution as the SDE.

Guided Diffusion Models

Guidance

• So far, we have focused on modeling just the data distribution $p(\boldsymbol{x})$. However, we are often also interested in learning conditional distribution $p(\boldsymbol{x}|y)$, enabling us to control the data we generated through conditioning information y. This is the backbone of DALL-E2 and Imagen.

• We can add arbitrary conditioning information y at each transition step as follows:

$$p(\boldsymbol{x}_{0:T}|y) = p(\boldsymbol{x}_T)\prod_{t=1}^{T} p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, y).$$

For example, y could be a low-resolution image to perform super-resolution on or physical properties in material sciences and biophysics.

• The two most popular forms of guidance are Classifier Guidance and Classifier-Free Guidance.

Classifier Guidance

- Consider learning $\nabla \log p(\boldsymbol{x}_t|y)$, the score of the conditional model, at arbitrary noise levels t.
- By Bayes rule, we have

$$\nabla \log p(\boldsymbol{x}_t|y) = \nabla \log \left(\frac{p(\boldsymbol{x}_t)p(y|\boldsymbol{x}_t)}{p(y)} \right) = \underbrace{\nabla \log p(\boldsymbol{x}_t)}_{\text{unconditional score}} + \underbrace{\nabla \log p(y|\boldsymbol{x}_t)}_{\text{adversarial gradient}}. \tag{1}$$

- In Classifier Guidance, the score of an unconditional diffusion model is learned as previously derived, alongside a classifier that takes in arbitrary noisy $\boldsymbol{x}_t$ and attempts to predict conditional information y. In sampling, the overall conditional score function is used for annealed Langevin dynamics.
- We can rescale the adversarial gradient of the noisy classifier to get

$$\nabla \log p(\boldsymbol{x}_t|y) = \nabla \log p(\boldsymbol{x}_t) + \gamma \nabla \log p(y|\boldsymbol{x}_t). \tag{2}$$

Classifier-Free Guidance

• Train a separate classifier in favor of an unconditional diffusion model and a conditional diffusion model. To derive the score function under Classifier-Free Guidance, we can first rearrange (1) to

$$\nabla \log p(y|\boldsymbol{x}_t) = \nabla \log p(\boldsymbol{x}_t|y) - \nabla \log p(\boldsymbol{x}_t). \tag{3}$$

• Substituting (3) into (2), we get

$$\begin{aligned} \nabla \log p(\boldsymbol{x}_t|y) &= \nabla \log p(\boldsymbol{x}_t) + \gamma(\nabla \log p(\boldsymbol{x}_t|y) - \nabla \log p(\boldsymbol{x}_t)) \\ &= \underbrace{\gamma \nabla \log p(\boldsymbol{x}_t|y)}_{\text{conditional score}} + \underbrace{(1-\gamma)\nabla \log p(\boldsymbol{x}_t)}_{\text{unconditional score}} \end{aligned} \tag{4}$$

• Learning two separate diffusion models is expensive, we can learn both conditional and unconditional diffusion models together as a singular conditional model: the unconditional diffusion model can be queried by replacing the conditioning information with fixed constant values, such as zeros.

AI for Science

Controllable generation for inverse problem solving

• Accelerate MRI and CT: Achieving even better performance than supervised or unrolled deep learning approaches, while being more robust to different measurement processes at test time.

• Inverse problems are the same as Bayesian inference. Let the forward process of generating $\boldsymbol{y}$ from $\boldsymbol{x}$ is given by the transition probability $p(\boldsymbol{y}|\boldsymbol{x})$, the inverse problem is to compute $p(\boldsymbol{x}|\boldsymbol{y})$.

• From Bayes' rule, we have $p(\boldsymbol{x}|\boldsymbol{y}) = p(\boldsymbol{x})p(\boldsymbol{y}|\boldsymbol{x})/\int p(\boldsymbol{x})p(\boldsymbol{y}|\boldsymbol{x})d\boldsymbol{x}$. Taking gradient w.r.t. $\boldsymbol{x}$ on both sides, leading to the following Bayes' rule for score functions

$$\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}|\boldsymbol{y}) = \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}) + \nabla_{\boldsymbol{x}} \log p(\boldsymbol{y}|\boldsymbol{x}). \tag{5}$$

• Through score matching, we can train a model to estimate the score function of the unconditional data distribution, i.e. $\boldsymbol{s}_\theta(\boldsymbol{x}) \approx \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x})$.

• This will allow us to compute the posterior score function $\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}|\boldsymbol{y})$ from the known forward process $p(\boldsymbol{y}|\boldsymbol{x})$ via (5), and sample from it with Langevin-type sampling.

ConfGF

References:

http://proceedings.mlr.press/v139/shi21b/shi21b.pdf

https://proceedings.neurips.cc/paper_files/paper/2021/file/a45a1d12ee0fb7f1f872ab91da18f899-Paper.pdf

• **Molecular Conformation Prediction:** Given an undirected graph $G = (V, E)$ with each node v_i associated with a nuclear charge Z_i and a 3D vector $\boldsymbol{r}_i$, indicating its atomic type and atomic coordinate, respectively. All distances between connected nodes $\boldsymbol{d} = \{d_{ij}\} \in \mathbb{R}^{|E|}$. We extend the original molecular graph by adding auxiliary edges, i.e., virtual bonds, between atoms that are 2 or 3 hops away. Given an extended molecular graph $G = (V, E)$, our goal is to learn a generative model to generate molecular conformations $\boldsymbol{R} = (\boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_{|V|})$ based on the molecular graph.

• We aim to model the score function of the p.d.f. w.r.t. conformation $s(\boldsymbol{R}) = \nabla_{\boldsymbol{R}} \log p(\boldsymbol{R}|G)$. As $\log p(\boldsymbol{R}|G)$ is roto-translation invariant w.r.t. conformations, so is the score function $s(\boldsymbol{R})$.

• To ensure $s(\boldsymbol{R})$ is roto-translational equivariant, ConfGF parameterize the log-density of coordinates $\boldsymbol{R}$ given the molecular graph G as $\log p_\theta(\boldsymbol{R}|G) := f_G \circ g_G(\boldsymbol{R}) = f_G(\boldsymbol{d})$, where $g_G : \mathbb{R}^{|V|\times 3} \to \mathbb{R}^{|E|}$ maps a set of atomic coordinates to a set of interatomic distances, and $f_G : \mathbb{R}^{|E|} \to \mathbb{R}$ is a GNN that estimates the negative energy of a molecule based on the interatomic distances $\boldsymbol{d}$ and the graph G.

• Estimating the gradient fields of atomic coordinates can be translated to estimating the gradient fields of interatomic distances, which can be estimated using score-based generative models.

• Note that the interatomic distances $\boldsymbol{d}$ are continuously differentiable w.r.t. atomic coordinates $\boldsymbol{R}$. Therefore, the two gradient fields, $\nabla_{\boldsymbol{R}} \log p_\theta(\boldsymbol{R}|G)$ and $\nabla_{\boldsymbol{d}} \log p_\theta(\boldsymbol{d}|G)$, are connected by chain rule:

$$\boldsymbol{s}_\theta(\boldsymbol{R})_i = \frac{\partial f_G(\boldsymbol{d})}{\partial \boldsymbol{r}_i} = \sum_{(i,j),e_{ij}\in E} \frac{\partial f_G(\boldsymbol{d})}{\partial d_{ij}} \cdot \frac{\partial d_{ij}}{\partial \boldsymbol{r}_i} = \sum_{j\in N(i)} \frac{1}{d_{ij}} \cdot \frac{\partial f_G(\boldsymbol{d})}{\partial d_{ij}} \cdot (\boldsymbol{r}_i - \boldsymbol{r}_j) = \sum_{j\in N(i)} \frac{1}{d_{ij}} \cdot \boldsymbol{s}_\theta(\boldsymbol{d})_{ij} \cdot (\boldsymbol{r}_i - \boldsymbol{r}_j),$$

where $\boldsymbol{s}_\theta(\boldsymbol{R})_i = \nabla_{\boldsymbol{r}_i} \log p_\theta(\boldsymbol{R}|G)$ and $\boldsymbol{s}_\theta(\boldsymbol{d})_{ij} = \nabla_{d_{ij}} \log p_\theta(\boldsymbol{d}|G)$.

• By the above observation, we first train a noise conditional score network $\boldsymbol{s}_\theta(\boldsymbol{d}, \sigma)$ to approximate $\nabla_{\tilde{\boldsymbol{d}}} \log q_\sigma(\tilde{\boldsymbol{d}}|G)$. We then calculate the gradient fields of atomic coordinates $\boldsymbol{s}_\theta(\boldsymbol{R}, \sigma)$ from $\boldsymbol{s}_\theta(\boldsymbol{d}, \sigma)$. Such designed score network $\boldsymbol{s}_\theta(\boldsymbol{R}, \sigma)$ enjoys roto-translation equivariance.

• Let $\{\sigma_i\}_{i=1}^L$ be a geometric progression with common ratio γ, i.e. $\sigma_i/\sigma_{i-1} = \gamma$. We define the perturbed conditional distance distribution $q_\sigma(\tilde{\boldsymbol{d}}|G)$ to be $\int p(\boldsymbol{d}|G)\mathcal{N}(\tilde{\boldsymbol{d}}|\boldsymbol{d}, \sigma_i^2 \boldsymbol{I})d\boldsymbol{d}$. We aim to learn a conditional score network to jointly estimate the scores of all perturbed distance distributions, i.e. $\forall \sigma \in \{\sigma_i\}_{i=1}^L : \boldsymbol{s}_\theta(\tilde{\boldsymbol{d}}, \sigma) \approx \nabla_{\tilde{\boldsymbol{d}}} \log q_\sigma(\tilde{\boldsymbol{d}}|G)$. Note that $\boldsymbol{s}_\theta(\tilde{\boldsymbol{d}}, \sigma) \in \mathbb{R}^{|E|}$ – Much higher-dimension.

• ConfGF circumvents directly modeling atomic coordinates using intermediate geometric variables such as atomic distances, bond, and torsion angles, which are roto-translational invariant.

• We embed node and edge attributes into low-dimensional feature spaces using feedforward networks

$$\boldsymbol{h}_i^0 = MLP(Z_i)\ \forall v_i \in V;\quad \boldsymbol{h}_{e_{ij}} = MLP(e_{ij}, d_{ij}),\quad \forall e_{ij} \in E.$$

Then use graph isomorphism network to compute node embeddings based on graph structures:

$$\boldsymbol{h}_i^l = MLP(\boldsymbol{h}_i^{l-1} + \sum_{j \in N(i)} ReLU(\boldsymbol{h}_j^{l+1} + \boldsymbol{h}_{e_{ij}})).$$

We compute the final edge embeddings by concatenating the corresponding node embeddings:

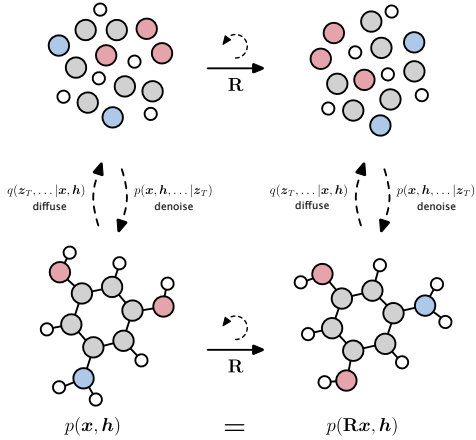
$$\boldsymbol{h}_{e_{ij}}^o = \boldsymbol{h}_i^N || \boldsymbol{h}_j^N || \boldsymbol{h}_{e_{ij}}.$$

• We parameterize the noise conditional network with $\boldsymbol{s}_\theta(\tilde{\boldsymbol{d}}, \sigma) = \boldsymbol{s}_\theta(\tilde{\boldsymbol{d}})/\sigma$ as follows

$$\boldsymbol{s}_\theta(\tilde{\boldsymbol{d}})_{ij} = MLP(\boldsymbol{h}_{e_{ij}}^o),\quad \forall e_{ij} \in E,$$

where $\boldsymbol{s}_\theta(\tilde{\boldsymbol{d}})$ is an unconditional score network, and the MLP maps edge embeddings to a scalar for each $e_{ij} \in E$. Note that $\boldsymbol{s}_\theta(\tilde{\boldsymbol{d}}, \sigma)$ is roto-translation invariant as we only use interatomic distances.

EDM



Reference: `https://arxiv.org/abs/2203.17003`

• EDM learns to denoise a diffusion process with an equivariant network that jointly operates on both continuous (atom coordinates) and categorical features (atom types).

• A conditional distribution $p(\boldsymbol{y}|\boldsymbol{x})$ is equivariant to the action of rotations or reflections when

$$p(\boldsymbol{y}|\boldsymbol{x}) = p(\boldsymbol{R}\boldsymbol{y}|\boldsymbol{R}\boldsymbol{x}), \quad \text{for all orthogonal } \boldsymbol{R}.$$

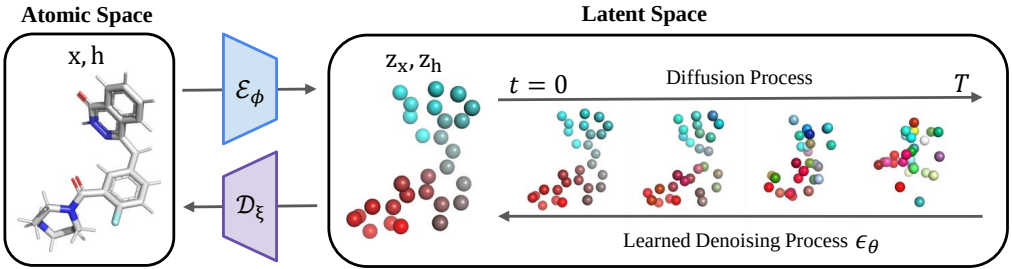
A distribution is invariant to $\boldsymbol{R}$ transformation if $p(\boldsymbol{y}) = p(\boldsymbol{R}\boldsymbol{y}), \quad$ for all orthogonal $\boldsymbol{R}$.

• If $\boldsymbol{x} \sim p(\boldsymbol{x})$ is invariant to a group and the transition probabilities of a Markov chain $\boldsymbol{y} \sim p(\boldsymbol{y}|\boldsymbol{x})$ are equivariant, then the marginal distribution of $\boldsymbol{y}$ at any time step is invariant to group transformations. It means that if $p(\boldsymbol{z}_T)$ is invariant and the neural network $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ is equivariant, then the marginal distribution $p(\boldsymbol{x})$ of the denoising model will be invariant.

• Parameterize diffusion model (the noise term) using EGNN.

• Problem: How to extend EDM to score-based diffusion models?

GeoLDM: Geometric Latent Diffusion Models for 3D Molecule Generation



Reference: `https://proceedings.mlr.press/v202/xu23n/xu23n.pdf`

• Each molecule is represented as points cloud $\mathcal{G} = (\boldsymbol{x}, \boldsymbol{h})$, where $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N) \in \mathbb{R}^{N\times 3}$ are the atom coordinates and $\boldsymbol{h} = (\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N) \in \mathbb{R}^{N\times d}$ are the node features, such as atomic type and charges.

Unconditional generation. With a collection of molecules $\mathcal{G}$, learn parameterized generative models $p_\theta(\mathcal{G})$ which can generate diverse and realistic molecules $\hat{\mathcal{G}}$ in 3D.

Controllable generation. With molecules $\mathcal{G}$ that has certain properties s, learn conditional generation models $p_\theta(\mathcal{G}|s)$, which can conduct controllable molecule generation given desired property value s.

• The learned likelihood should be invariant to roto-translation.

• Geometric autoencoder: The encoder $\mathcal{E}_\phi$ encodes $\mathcal{G}$ into latent domain $\boldsymbol{z} = \mathcal{E}_\phi(\boldsymbol{x}, \boldsymbol{h})$ and the decoder $\mathcal{D}_\xi$ learns to decode $\boldsymbol{z}$ back to data domain $\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{h}} = \mathcal{D}_\xi(\boldsymbol{z})$.

• Parameterization of latent space as invariant scalar-valued features is challenging. Geometric AE requires an additional function ψ to represent appropriate group actions for encoding, and aligning output and input positions for decoding, to solve the reconstruction task.

• GeoLDM incorporates equivariance into $\mathcal{E}$ and $\mathcal{D}$ by constructing latent features as point-structured variables $\boldsymbol{z} = (\boldsymbol{z}_x, \boldsymbol{z}_h) \in \mathbb{R}^{N\times(3+k)}$, which holds 3D equivariant and kD invariant latent features $\boldsymbol{z}_x$ and $\boldsymbol{z}_h$ for each node. GeoLDM implements it by parameterizing $\mathcal{E}$ and $\mathcal{D}$ with EGNN, which extracts both invariant and equivariant embeddings.

• The latent space $\boldsymbol{z}$ contains not only scalars (invariant features) $\boldsymbol{z}_h$ but also tensors (equivariant features) $\boldsymbol{z}_x$. This requires the distribution of latent DMs to satisfy $p_\theta(\boldsymbol{z}_x, \boldsymbol{z}_h) = p_\theta(\boldsymbol{R}\boldsymbol{z}_x, \boldsymbol{z}_h),\ \forall \boldsymbol{R}$, and GeoLDM implements latent DMs using EGNN.

• For the number of nodes N, we need to sample different numbers N for generating molecules of different sizes. We use the distribution $p(N)$ of molecular sizes on the training set. Then for generation, we can first sample $N \sim p(N)$ and then generate latent variables and node features in size N.

• DMs are capable of controllable generation with given conditions s, by modeling conditional distributions $p(\boldsymbol{z}|s)$. This in DMs can be implemented with conditional denoising networks $\epsilon_\theta(\boldsymbol{z}, t, s)$, with the critical difference that it takes additional input s.

SyMat: Towards Symmetry-Aware Generation of Periodic Materials

Reference: https://arxiv.org/abs/2307.02707

• SyMat generates atom types and lattices of materials by generating atom type sets, lattice lengths, and lattice angles with VAE. SyMat further employs a symmetry-aware score-based diffusion model to generate atom coordinates of materials.

• Assume there are n atoms in a unit cell of the periodic material $\boldsymbol{M} = (\boldsymbol{A}, \boldsymbol{P}, \boldsymbol{L})$, where $\boldsymbol{A}$, $\boldsymbol{P} \in \mathbb{R}^{3\times n}$, and $\boldsymbol{L} \in \mathbb{R}^{3\times 3}$ are the atom type vector, coordinate matrix, and lattice matrix, resp.

• We are given a dataset $\mathcal{M} = \{\boldsymbol{M}_j\}_{j=1}^{m}$ where each $\boldsymbol{M}_j$ is assumed to be sampled from the distribution $p(\cdot)$, which is invariant to the following symmetry transformation:

> **Permutation.** If we permute $\boldsymbol{A}$ to get $\boldsymbol{A}'$ and use the same order to permute the column vectors of $\boldsymbol{P}$ to obtain $\boldsymbol{P}'$. Let $\boldsymbol{M}' = (\boldsymbol{A}', \boldsymbol{P}', \boldsymbol{L})$, then $p(\boldsymbol{M}) = p(\boldsymbol{M}')$.

> **Rotation and translation.** For any $\boldsymbol{M} = (\boldsymbol{A}, \boldsymbol{P}, \boldsymbol{L})$, we can rotate and translate its atom positions to obtain $\boldsymbol{M}' = (\boldsymbol{A}, \boldsymbol{P}', \boldsymbol{L}')$. Then $p(\boldsymbol{M}) = p(\boldsymbol{M}')$.

> **Periodic transformation.** A periodically equivalent coordinate matrix $\boldsymbol{P}^K$ of $\boldsymbol{P}$ can be obtained by periodic transformation as $\boldsymbol{P}^K = \boldsymbol{P} + \boldsymbol{L}\boldsymbol{K}$, where $\boldsymbol{K} \in \mathbb{Z}^{3\times n}$. Let $\boldsymbol{M} = (\boldsymbol{A}, \boldsymbol{P}, \boldsymbol{L})$ and $\boldsymbol{M}^K = (\boldsymbol{A}, \boldsymbol{P}^K, \boldsymbol{L})$ then $p(\boldsymbol{M}) = p(\boldsymbol{M}^K)$.

• We learn a generative model $p_\theta(\cdot)$ from the given dataset $\mathcal{M}$ to capture the real data distribution $p(\cdot)$, and the learned generation model can generate a valid material structure $\boldsymbol{M}$ using $p_\theta(\boldsymbol{M})$.

SyMat: Atom Type and Lattice Generation

• Using VAE to directly generate and capture the distribution of $\boldsymbol{A}$ and $\boldsymbol{L}$ needs to include permutation invariance of $\boldsymbol{A}$ and rotation invariance of $\boldsymbol{L}$. SyMat transforms $\boldsymbol{A}$ and $\boldsymbol{L}$ to items that are invariant to symmetry transformations and makes these items the direct generation targets of the VAE model.

(1) We count the number of atoms for every type and represent $\boldsymbol{A}$ with an unordered k-element atom type set $\boldsymbol{c} = \{(e_1, n_1), \ldots, (e_k, n_k)\}$. We see that $\boldsymbol{c}$ is invariant to any permutation on $\boldsymbol{A}$.

(2) We use six rotation-invariant items for lattice $\boldsymbol{L}$ generation, including three lattice lengths $\boldsymbol{l} = [l_1, l_2, l_3]$ and three lattice angles $\phi = [\phi_{12}, \phi_{13}, \phi_{23}]$, where ϕ_{ij} is the angle between $\boldsymbol{l}_i$ and $\boldsymbol{l}_j$.

• The VAE model used for generating $\boldsymbol{c}, \boldsymbol{l}$, and ϕ consists of an encoder and a decoder. The encoder uses symmetry-aware SphereNet that takes a material $\boldsymbol{M} = (\boldsymbol{A}, \boldsymbol{P}, \boldsymbol{L})$ as inputs and outputs a latent variable $\boldsymbol{z}^{\boldsymbol{A}} \in \mathbb{R}^d$ and a latent variable $\boldsymbol{z}^{\boldsymbol{L}} \in \mathbb{R}^f$.

• For any material $\boldsymbol{M} = (\boldsymbol{A}, \boldsymbol{P}, \boldsymbol{L})$ in $\mathcal{M}$, the encoder first maps it to latent variable $\boldsymbol{z}^{\boldsymbol{A}}$ and $\boldsymbol{z}^{\boldsymbol{L}}$. Afterwards, the decoder uses $\boldsymbol{z}^{\boldsymbol{A}}$ and $\boldsymbol{z}^{\boldsymbol{L}}$ to reconstruct the exact $\boldsymbol{c}, \boldsymbol{l}$, and ϕ obtained from $\boldsymbol{A}$ and $\boldsymbol{L}$.

SyMat: Coordinate Generation

• SyMat generates the coordinate matrix $\boldsymbol{P}$ conditioned on $\boldsymbol{A}$ and $\boldsymbol{L}$ using a score-based diffusion model with score matrix $\nabla_{\boldsymbol{P}} \log p(\boldsymbol{P}|\boldsymbol{A}, \boldsymbol{L})$.

• $\nabla_{\boldsymbol{P}} \log p(\boldsymbol{P}|\boldsymbol{A}, \boldsymbol{L})$ satisfies:

(1) $\nabla_{\boldsymbol{P}} \log p(\boldsymbol{P}|\boldsymbol{A}, \boldsymbol{L})$ is equivariant to permutations on $\boldsymbol{A}$ and $\boldsymbol{P}$

(2) $\nabla_{\boldsymbol{QP}+\boldsymbol{b}\mathbf{1}^\top} \log p(\boldsymbol{QP} + \boldsymbol{b}\mathbf{1}^\top|\boldsymbol{A}, \boldsymbol{QL}) = \boldsymbol{Q}\nabla_{\boldsymbol{P}} \log p(\boldsymbol{P}|\boldsymbol{A}, \boldsymbol{L})$ for any orthogonal matrix $\boldsymbol{Q}$ and $\boldsymbol{b}$

(3) $\nabla_{\boldsymbol{P}+\boldsymbol{LK}} \log p(\boldsymbol{P} + \boldsymbol{LK}|\boldsymbol{A}, \boldsymbol{L}) = \nabla_{\boldsymbol{P}} \log p(\boldsymbol{P}|\boldsymbol{A}, \boldsymbol{L})$ holds for any $\boldsymbol{K} \in \mathbb{Z}^{3\times n}$.

• We formulate $\nabla_{\boldsymbol{P}} \log p(\boldsymbol{P}|\boldsymbol{A}, \boldsymbol{L})$ as a score function of edge distances in a graph created by a multi graph method. For $\boldsymbol{M}$ with coordinates $\boldsymbol{P} = [\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n]$ and lattice matrix $\boldsymbol{L}$, multi graph method produces an n-node undirected graph on $\boldsymbol{M}$ – the node i corresponds to the i-th atom in the unit cell, and an edge is added between any two nodes i, j if one of the interatomic distances between them is within cutoff r.

• The interatomic distances between i, j are the distances between their corresponding atoms in the complete infinite structure of $\boldsymbol{M}$, including both the distance within the unit cell and the distance crossing the unit cell boundary. Formally, the set of all edges in the graph constructed on $\boldsymbol{M}$ can be written as $\boldsymbol{E}(\boldsymbol{M}) = \{(i, j, \boldsymbol{k}) : \|\boldsymbol{p}_i + \boldsymbol{Lk} - \boldsymbol{p}_j\|_2 \leq r, \boldsymbol{k} \in \mathbb{Z}^3, 1 \leq i, j \leq n\}$.

SyMat: Coordinate Generation

• Let $d_{i,j,\boldsymbol{k}} = \|\boldsymbol{p}_i + \boldsymbol{L}\boldsymbol{k} - \boldsymbol{p}_j\|_2$ be the distance of the edge $(i, j, \boldsymbol{k})$, we consider $\log p(\boldsymbol{P}|\boldsymbol{A}, \boldsymbol{L})$ as a function of the distances of all edges in $E(\boldsymbol{M})$. Denoting $\boldsymbol{s}_i$ as the score function of $\boldsymbol{p}_i$, then $\nabla_{\boldsymbol{P}} \log p(\boldsymbol{P}|\boldsymbol{A}, \boldsymbol{L}) = [\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n]$. From the chain rule, we can calculate $\boldsymbol{s}_i$ as

$$\boldsymbol{s}_i = \sum_{(j,\boldsymbol{k})\in\mathcal{N}(i)} \nabla_{d_{i,j,\boldsymbol{k}}} \log p(\boldsymbol{P}|\boldsymbol{A}, \boldsymbol{L}) \cdot \nabla_{\boldsymbol{p}_i} d_{i,j,\boldsymbol{k}} = \sum_{(j,\boldsymbol{k})\in\mathcal{N}(i)} s_{i,j,\boldsymbol{k}} \cdot \frac{\boldsymbol{p}_i + \boldsymbol{L}\boldsymbol{k} - \boldsymbol{p}_j}{d_{i,j,\boldsymbol{k}}},$$

where $s_{i,j,\boldsymbol{k}} = \nabla_{d_{i,j,\boldsymbol{k}}} \log p(\boldsymbol{P}|\boldsymbol{A}, \boldsymbol{L})$ is the score function of the distance $d_{i,j,\boldsymbol{k}}$. $\nabla_{\boldsymbol{P}} \log p(\boldsymbol{P}|\boldsymbol{A}, \boldsymbol{L})$ can be approximated by first approximating $s_{i,j,\boldsymbol{k}}$ for every edge in the multi-graph representation of $\boldsymbol{M}$, then calculating each column vector of $\nabla_{\boldsymbol{P}} \log p(\boldsymbol{P}|\boldsymbol{A}, \boldsymbol{L})$.

• If $s_{i,j,\boldsymbol{k}}$ is invariant to all symmetry transformations described before for every edge $(i, j, \boldsymbol{k})$, the score matrix $\nabla_{\boldsymbol{P}} \log p(\boldsymbol{P}|\boldsymbol{A}, \boldsymbol{L})$ satisfies the desired symmetry.

• SyMat uses SphereNet as the score model $\boldsymbol{s}_\theta(\cdot)$ to approximate $\nabla_{\boldsymbol{P}} \log p(\boldsymbol{P}|\boldsymbol{A}, \boldsymbol{L})$. $\boldsymbol{s}_\theta(\cdot)$ takes the multi-graph representation of $\boldsymbol{M}$ as input and outputs $o_{i,j,\boldsymbol{k}}$ as the approximated $s_{i,j,\boldsymbol{k}}$ at every edge $(i, j, \boldsymbol{k})$ in the input graph.

DiffCSP: Crystal Structure Prediction by Joint Equivariant Diffusion

Reference: `https://arxiv.org/abs/2309.04475`

• **Representation of crystal structures.** A 3D crystal can be represented as the infinite periodic arrangement of atoms in 3D space, and the smallest repeating unit is called a unit cell. A unite cell can be defined by a triplet $\mathcal{M} = (\boldsymbol{A}, \boldsymbol{X}, \boldsymbol{L})$, where $\boldsymbol{A} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_N] \in \mathbb{R}^{h \times N}$ denotes the list of the one-hot representation of atom types, $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N] \in \mathbb{R}^{3 \times N}$ consists of Cartesian coordinates of the atoms, and $\boldsymbol{L} = [\boldsymbol{l}_1, \boldsymbol{l}_2, \boldsymbol{l}_3] \in \mathbb{R}^{3 \times 3}$ represents the lattice matrix consisting three basic vectors to describe the periodicity of the crystal. The infinite periodic crystal structure is represented by

$$\{(\boldsymbol{a}_i', \boldsymbol{x}_i') | \boldsymbol{a}_1' = \boldsymbol{a}_i, \boldsymbol{x}_i' = \boldsymbol{x}_i + \boldsymbol{L}\boldsymbol{k}, \forall \boldsymbol{k} \in \mathbb{Z}^{3 \times 1}\},$$

where the j-th element of the integral vector $\boldsymbol{k}$ denotes the integral 3D translation in units of $\boldsymbol{l}_j$.

Fractional coordinate system. The Cartesian coordinate system $\boldsymbol{X}$ leverages three standard orthogonal bases as the coordinate axes. The fractional coordinate system uses the lattices $(\boldsymbol{l}_1, \boldsymbol{l}_2, \boldsymbol{l}_3)$ as the bases. A point represented by fractional coordinate vector $\boldsymbol{f} = [f_1, f_2, f_3]^\top \in [0, 1)^3$ corresponds to Cartesian vector $\boldsymbol{x} = \sum_{i=1}^{3} f_i \boldsymbol{l}_i$. Using a fractional coordinate system, we denote the crystal by $\mathcal{M} = (\boldsymbol{A}, \boldsymbol{F}, \boldsymbol{L})$, where the fractional coordinates of all atoms in a cell compose the matrix $\boldsymbol{F} = [0, 1)^{3 \times N}$.

Task. CSP predicts for each unite cell the lattice matrix $\boldsymbol{L}$ and the fractional matrix $\boldsymbol{F}$ given its chemical composition $\boldsymbol{A}$, namely, learning the conditional distribution $p(\boldsymbol{L}, \boldsymbol{F} | \boldsymbol{A})$.

• Generating crystal structures requires not only modeling the distribution of the atom coordinates within every cell but also inferring how their bases (a.k.a. lattice vectors) are placed in 3D space.

• Periodic E(3) invariance: Any E(3) transformation of the crystal coordinates does not change the physical law and thus keeps the crystal distribution invariant. Moreover, crystal structure prediction (CSP) exhibits unique challenges due to the periodicity of the atom arrangement in crystals.

• DiffCSP jointly generates the lattice and atom coordinates for each crystal using a periodic E(3) equivariant diffusion model, to model the crystal geometry.

• DiffCSP leverages the fractional coordinate systems other than the Cartesian system, which encodes periodicity intrinsically. The fractional representation not only allows us to consider Wrapped Normal (WN) distribution to better model the periodicity, but also facilitates the design of the denoising model via the Fourier transformation.

• The denoising process in diffusion models acts like a force field that drives the atom coordinates towards the energy local minimum and thus is able to increase stability.

- We consider three types of symmetries in $p(\boldsymbol{L}, \boldsymbol{F}|\boldsymbol{A})$:

  > (Permutation invariance) For any permutation $\boldsymbol{P} \in S_N$, $p(\boldsymbol{L}, \boldsymbol{F}|\boldsymbol{A}) = p(\boldsymbol{L}, \boldsymbol{FP}|\boldsymbol{AP})$, i.e. changing the order of atoms will not change the distribution.

  > (O(3) invariance) For any orthogonal transformation $\boldsymbol{Q} \in \mathbb{R}^{3\times 3}$, $p(\boldsymbol{QL}, \boldsymbol{F}|\boldsymbol{A}) = p(\boldsymbol{L}, \boldsymbol{F}|\boldsymbol{A})$.

  > (Periodic translation invariance) For $\forall \boldsymbol{t} \in \mathbb{R}^{3\times 1}$, $p(\boldsymbol{L}, w(\boldsymbol{F} + \boldsymbol{t}1^\top)|\mathrm{A}) = \mathrm{p}(\mathrm{L}, \mathrm{F}|\mathrm{A})$, where the function $w(\boldsymbol{F}) = \boldsymbol{F} - \lceil \boldsymbol{F} \rceil \in [0, 1)^{3\times N}$ returns the fractional part of each element in $\boldsymbol{F}$.

- The permutation invariance is encapsulated by using GNNs. We focus on periodic E(3) invariance.

- With the help of the fractional system, the periodic E(3) invariance is made tractable by fulfilling O(3) invariance on $\boldsymbol{L}$ and periodic translation invariance on $\boldsymbol{F}$.

- $\boldsymbol{L}$ is a continuous variable; we use equivariant DDPM to accomplish the generation.

• The domain of fractional coordinates $[0,1)^{3\times N}$ induced by periodicity. DDPM is not suited to generate $\boldsymbol{F}$ – the normal distribution used in DDPM cannot model the cyclical and bounded domain.

• We use a score matching-based framework along with Wrapped Normal (WN) distribution to fit the specificity here. During the forward process, we first sample each column of $\boldsymbol{\epsilon_F} \in \mathbb{R}^{3\times N}$ from $\mathcal{N}(0, \mathrm{I})$, and then acquire $\boldsymbol{F}_t = w(\boldsymbol{F}_0 + \sigma_t \boldsymbol{\epsilon_F})$. This truncated sampling implies the WN transition:

$$q(\boldsymbol{F}_t|\boldsymbol{F}_0) \propto \sum_{\boldsymbol{Z}\in\mathbb{Z}^{3\times N}} \exp\Big(-\frac{\|\boldsymbol{F}_t - \boldsymbol{F}_0 + \boldsymbol{Z}\|_F^2}{2\sigma_t^2}\Big).$$

This process ensures the probability distribution over $[z, z+1)^{3\times N}$ for any integer z to be the same to keep the crystal periodicity. $q(\boldsymbol{F}_t|\boldsymbol{F}_0)$ is periodic translation equivariant, and approaches a uniform distribution $U(0,1)$ if σ_T is sufficiently large.

• The marginal distribution $p(\boldsymbol{F}_0)$ is periodic translation invariant if $\hat{\boldsymbol{\epsilon}}_{\boldsymbol{F}}(\mathcal{M}_t, t)$ is periodic translation invariant, i.e. $\hat{\boldsymbol{\epsilon}}_{\boldsymbol{F}}(\boldsymbol{L}_t, \boldsymbol{F}_t, \boldsymbol{A}, t) = \hat{\boldsymbol{\epsilon}}_{\boldsymbol{F}}(\boldsymbol{L}_t, w(\boldsymbol{F}_t + \boldsymbol{t}\mathbf{1}^\top), \boldsymbol{A}, t), \forall \boldsymbol{t} \in \mathbb{R}^3$.

• Training objective: $\mathcal{L}_{\boldsymbol{F}} = \mathbb{E}_{\boldsymbol{F}_t\sim q(\boldsymbol{F}_t|\boldsymbol{F}_0), t\sim U(1,T)}\Big[\lambda_t \|\nabla_{\boldsymbol{F}_t} \log q(\boldsymbol{F}_t|\boldsymbol{F}_0) - \hat{\boldsymbol{\epsilon}}_{\boldsymbol{F}}(\mathcal{M}_t, t)\|_2^2\Big]$,