Diffusion Models: Principles and Applications

by Sinchan Ghosh

IIT Bombay

July 1, 2025

Generative AI – A New Generation of AI Systems

Realastic Images

Natural Beauty

Vintage Photographs

These systems are "creative": they generate new objects.

Applications of Diffusion Models

- **Image Generation:** Used in state-of-the-art models like **DALL-E, Midjourney, Stable Diffusion, and Imagen** for synthesizing high-quality, diverse, and realistic images from textual or visual prompts.
- **Image Editing:** Enables precise **inpainting, outpainting, and style transfer**—allowing seamless modification and enhancement of existing images.
- **Video Generation:** Powers models like **Sora and Make-A-Video** to generate coherent video sequences from text or image inputs, marking a breakthrough in generative video modeling.
- **3D Content Generation:** Translates text prompts into 3D structures or meshes, enabling applications in gaming, AR/VR, and design automation.
- **Other Domains:** Includes audio synthesis (e.g., **text-to-speech, music**), **drug and molecule discovery** in computational chemistry, and **data-driven medical imaging and diagnostics**.

Today's Agenda

- **Overview of Generative Models**
- **What are Diffusion Models?**
  - Forward noising process
  - Reverse generative process
- **Training and Sampling**
  - Training objectives (e.g., ELBO)
  - Sampling method
- **Other Equivalent Formulations**
  - Score-Based Generative Modelling
- **Diffusions as continuous-time SDE**
- **Types and Variants**
- **Challenges and Future Directions**

Generative Modeling: The Landscape

What does it mean to "generate" an image?

- **Generative Modeling as Sampling:** We can translate the problem of e.g., "how to generate an image of a dog?" into the more precise problem of sampling from a probability distribution of dogs.
- **Goal:** Given observed samples x_0 from a distribution of interest, the goal of **generative modeling** is to learn to model its true data distribution $p(x_0)$ and once learned we can generate new samples from it at our will.

The core idea of generation

- The object we want to generate (e.g., image, video, audio) is **represented as a vector $\mathbf{x_0} \in \mathbb{R}^d$** potentially after flattening.
- Generation is modeled as **sampling from the data distribution**: $\mathbf{x_0} \sim p_{\text{data}}$. Therefore, how "good" an image/video/audio fits - a rather subjective statement- is replaced by how "likely" it is under the data distribution p_{data}.
- The true data distribution p_{data} is unknown, we only have access to a dataset $\{\mathbf{x_0^{(1)}}, \mathbf{x_0^{(2)}}, \ldots, \mathbf{x_0^{(N)}}\} \sim p_{\text{data}}$ as a proxy. So, all we need to do is somehow **come up with a way to draw samples from it without knowing the actual distribution**.
- How can we do that? Well, for this we assume access to a simple distribution p_{init} (e.g., $\mathcal{N}(0, \mathbf{I})$) from which we can easily sample. The goal is to **learn a transformation that maps samples from p_{init} to p_{data}**.

Overview of Generative Models

There are many types of generative models which have shown great success in generating high-quality samples, but each has some limitations of its own.

- **GANs**: High-quality samples but hard to train and less diversity in generation due to their adversarial training nature
- **VAEs**: Stable training but blurry outputs and relies on a surrogate objective (called ELBO)
- **Flows**: These models have to use specialized architectures to construct reversible transform
- **Diffusion Models**: known for their stability during training, high-quality outputs, and increasing dominance as the state-of-the-art in generative modeling. Unlike VAE or flow models, diffusion models are learned with a fixed procedure.

Overview of Generative Models

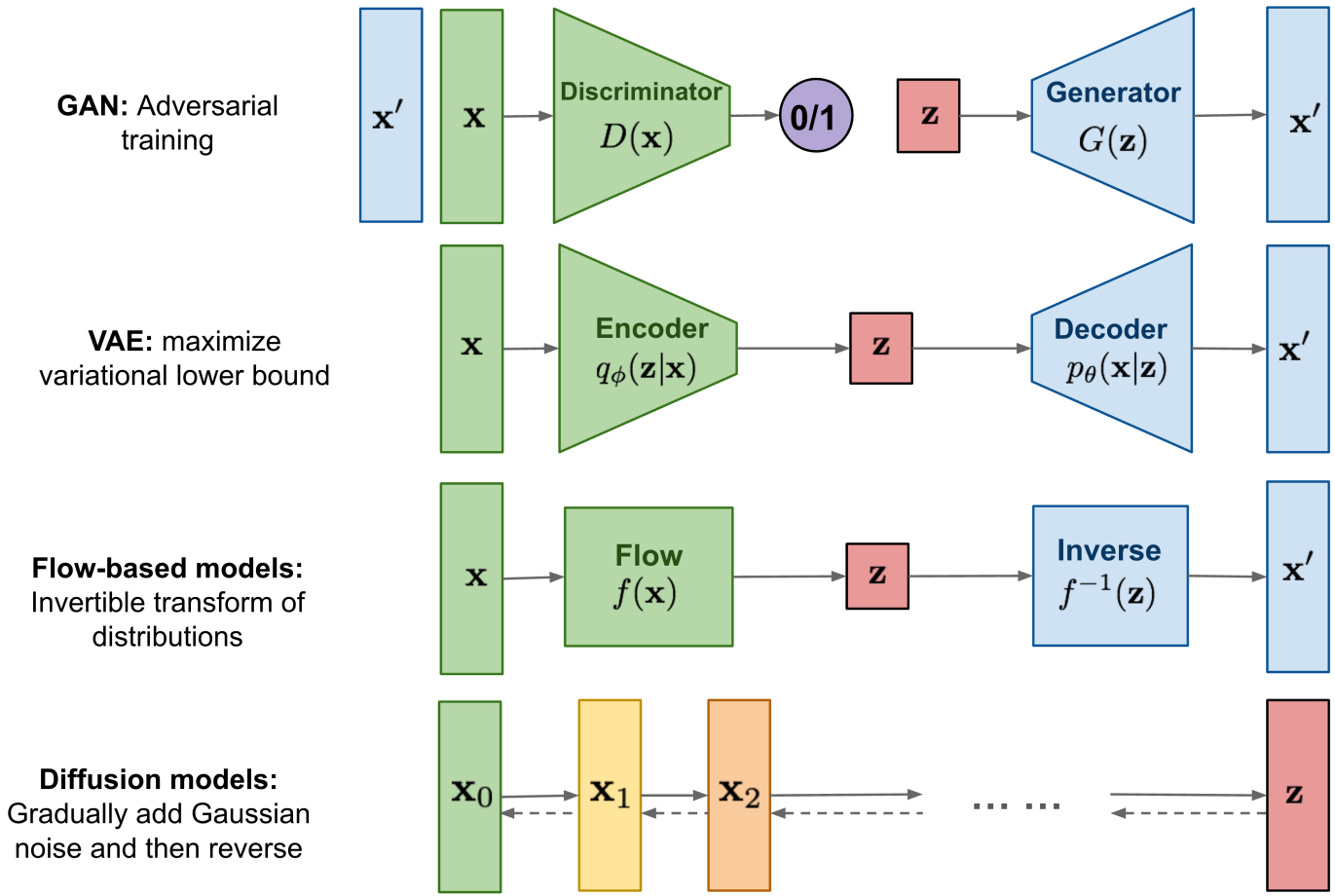


Figure: Various Generative Models

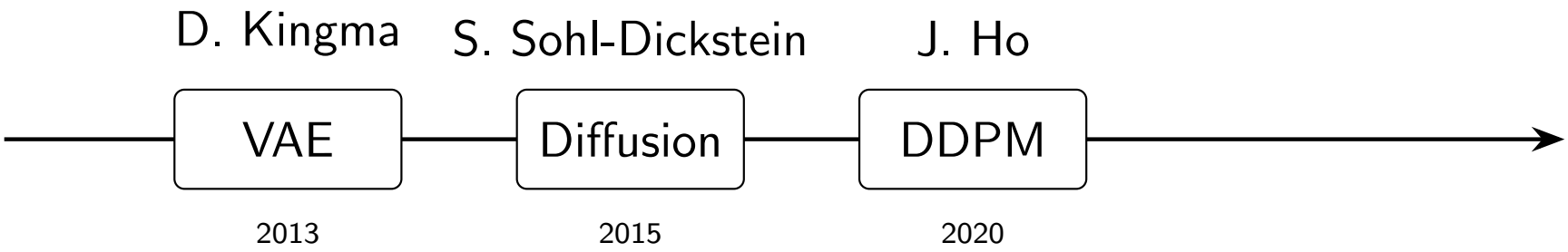
What are Diffusion Models?

Diffusion models are **inspired by non-equilibrium thermodynamics**. They **define a Markov chain of diffusion steps to slowly add random noise to the data and then learn to reverse the diffusion proces**.

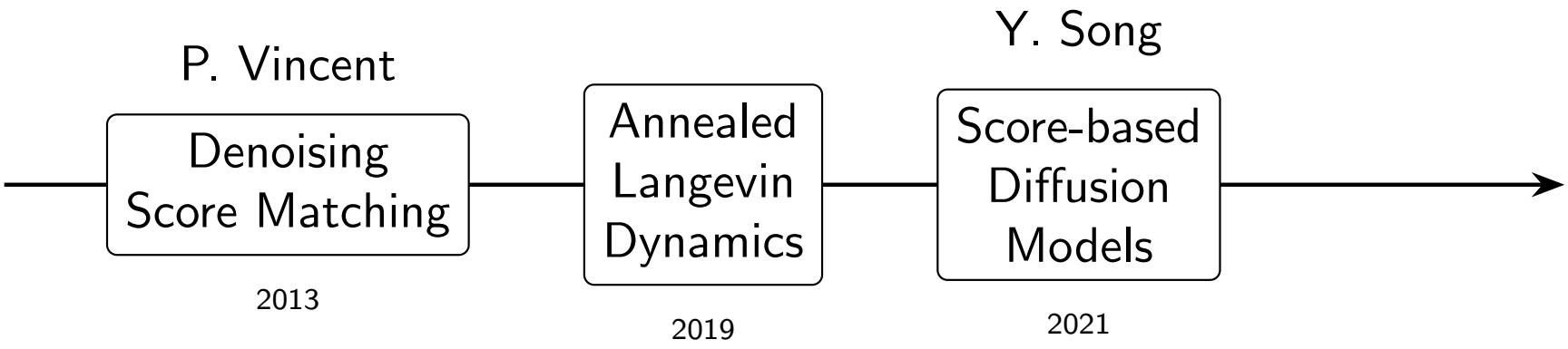
Several diffusion-based generative models have been proposed with similar ideas underneath. There are many ways to understand them, each shedding light on different aspects. Some of them include:

- **Diffusion Probabilistic Models** (Sohl-Dickstein et al., 2015),
- **Noise-Conditioned Score Networks** (NCSN; Yang Ermon, 2019),
- **Denoising Diffusion Probabilistic Models** (DDPM; Ho et al., 2020).

What are Diffusion Models?

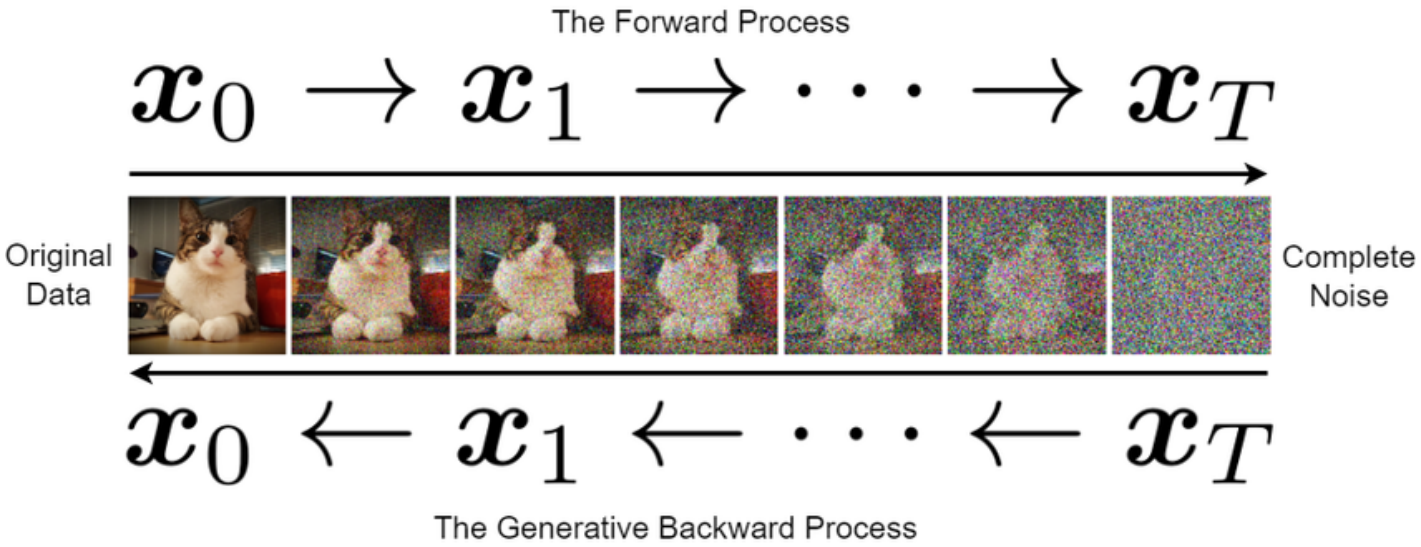


They both describe the same object!



What are Diffusion Models?

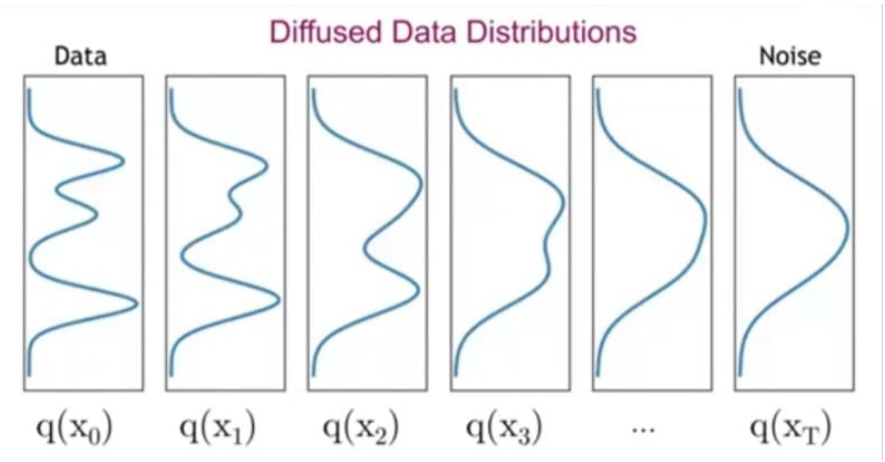
However, despite having differences in their formulations, the underlying concept remains the same: All of them **gradually add noise to an image and then learn to reverse the process by predicting noise** so that during generation process they can produce new realistic images starting from pure noise.



Forward Diffusion Process

- **Gradually adds Gaussian noise to data:**

Given a data point $\mathbf{x}_0 \sim q(\mathbf{x})$, **we define a forward diffusion process that progressively adds Gaussian noise in T steps**. This produces a sequence of noisy samples $\mathbf{x}_1, \ldots, \mathbf{x}_T$, with noise controlled by a variance schedule $\{\beta_t \in (0,1)\}_{t=1}^{T}$.



$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I})$$

As t increases, $\mathbf{x}_0$ gradually loses its structure. Eventually, as $T \to \infty$, the distribution of $\mathbf{x}_T$ approaches an **isotropic Gaussian**.

Forward Diffusion Process

- **Direct sampling from original data:**

  A nice property of the forward process is that we can sample $\mathbf{x}_t$ at any arbitrary timestep t in closed form using the reparameterization trick. Define $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$. Then:

$$\begin{aligned}
\mathbf{x}_t &= \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{t-1} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\bar{\boldsymbol{\epsilon}}_{t-2} \\
&\vdots \\
&= \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}
\end{aligned}$$

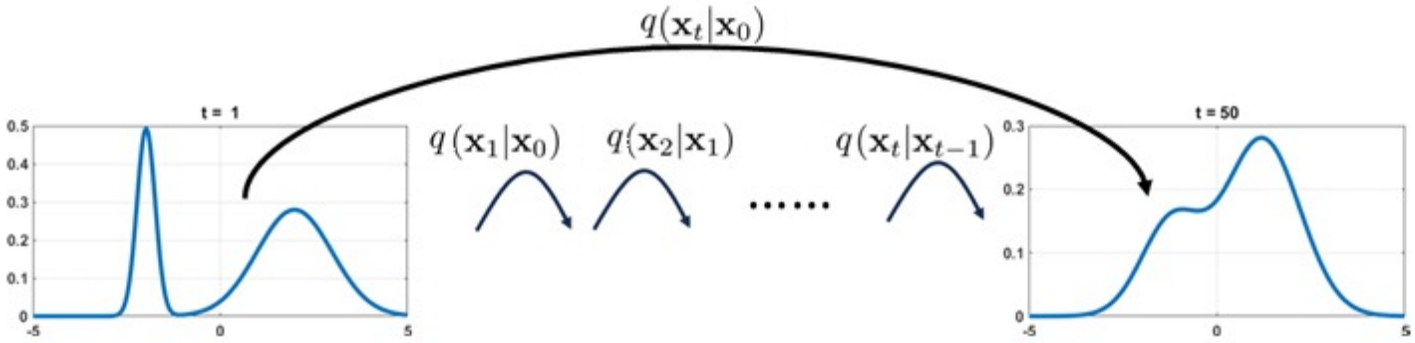
  where $\boldsymbol{\epsilon}_{t-1}, \boldsymbol{\epsilon}_{t-2}, \ldots \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

Therefore, $\quad q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}_t;\ \sqrt{\bar{\alpha}_t}\mathbf{x}_0,\ (1-\bar{\alpha}_t)\mathbf{I}\right)$

Forward Diffusion Process

- **Sampling at an arbitrary timestep:**

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$$



The difference between $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ and $q(\mathbf{x}_t|\mathbf{x}_0)$.

- **Variance scheduling:** As the sample gets noisier, we use larger noise steps. Hence, $\beta_1 < \beta_2 < \cdots < \beta_T$, which implies $\bar{\alpha}_1 > \cdots > \bar{\alpha}_T$.
- **Complete Forward Process:**

$$q(\mathbf{x}_{1:T} \mid \mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t \mid \mathbf{x}_{t-1})$$

Forward Diffusion Process

Why $\sqrt{\alpha_t}$ and $1-\alpha_t$?

Suppose that $q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t \mid a\mathbf{x}_{t-1}, b^2\mathbf{I})$ for some constants a and b. If we want to choose a and b such that the distribution of $\mathbf{x}_t$ will become $\mathcal{N}(0, \mathbf{I})$, then it is necessary that:

$$a = \sqrt{\alpha_t} \qquad \text{and} \qquad b = \sqrt{1-\alpha_t}.$$

$$\begin{aligned}
\mathbf{x}_t &= a\mathbf{x}_{t-1} + b\boldsymbol{\epsilon}_{t-1} && \text{where} \quad \boldsymbol{\epsilon}_{t-1} \sim \mathcal{N}(0, \mathbf{I}) \\
&= a(a\mathbf{x}_{t-2} + b\boldsymbol{\epsilon}_{t-2}) + b\boldsymbol{\epsilon}_{t-1} && (\text{substitute } \mathbf{x}_{t-1} = a\mathbf{x}_{t-2} + b\boldsymbol{\epsilon}_{t-2}) \\
&= a^2\mathbf{x}_{t-2} + ab\boldsymbol{\epsilon}_{t-2} + b\boldsymbol{\epsilon}_{t-1} && (\text{regroup terms}) \\
&\vdots \\
&= a^t\mathbf{x}_0 + b\underbrace{\left[\boldsymbol{\epsilon}_{t-1} + a\boldsymbol{\epsilon}_{t-2} + a^2\boldsymbol{\epsilon}_{t-3} + \cdots + a^{t-1}\boldsymbol{\epsilon}_0\right]}_{\stackrel{\text{def}}{=} \mathbf{w}_t}
\end{aligned}$$

Forward Diffusion Process

$$\begin{aligned}\text{Cov}(\mathbf{w}_t) &\overset{\text{def}}{=} \mathbb{E}[\mathbf{w}_t \mathbf{w}_t^\top] \\ &= b^2 \left(\text{Cov}(\boldsymbol{\epsilon}_{t-1}) + a^2 \text{Cov}(\boldsymbol{\epsilon}_{t-2}) + \cdots + a^{2(t-1)} \text{Cov}(\boldsymbol{\epsilon}_0) \right) \\ &= b^2 (1 + a^2 + a^4 + \cdots + a^{2(t-1)}) \mathbf{I} \\ &= b^2 \cdot \frac{1 - a^{2t}}{1 - a^2} \cdot \mathbf{I}\end{aligned}$$

As $t \to \infty$, $a^t \to 0$ for any $0 < a < 1$. Therefore, at the limit when $t = \infty$,

$$\lim_{t \to \infty} \text{Cov}[\mathbf{w}_t] = \frac{b^2}{1 - a^2} \mathbf{I}.$$

So, if we want $\lim_{t \to \infty} \text{Cov}[\mathbf{w}_t] = \mathbf{I}$ (so that the distribution of $\mathbf{x}_t$ will approach $\mathcal{N}(0, \mathbf{I})$), then we need

$$1 = \frac{b^2}{1 - a^2},$$

or equivalently $b = \sqrt{1 - a^2}$. Now, since we let $a = \sqrt{\alpha}$, so $b = \sqrt{1 - \alpha}$. This will give us:

$$\mathbf{x}_t = \sqrt{\alpha} \mathbf{x}_{t-1} + \sqrt{1 - \alpha} \boldsymbol{\epsilon}_{t-1}.$$

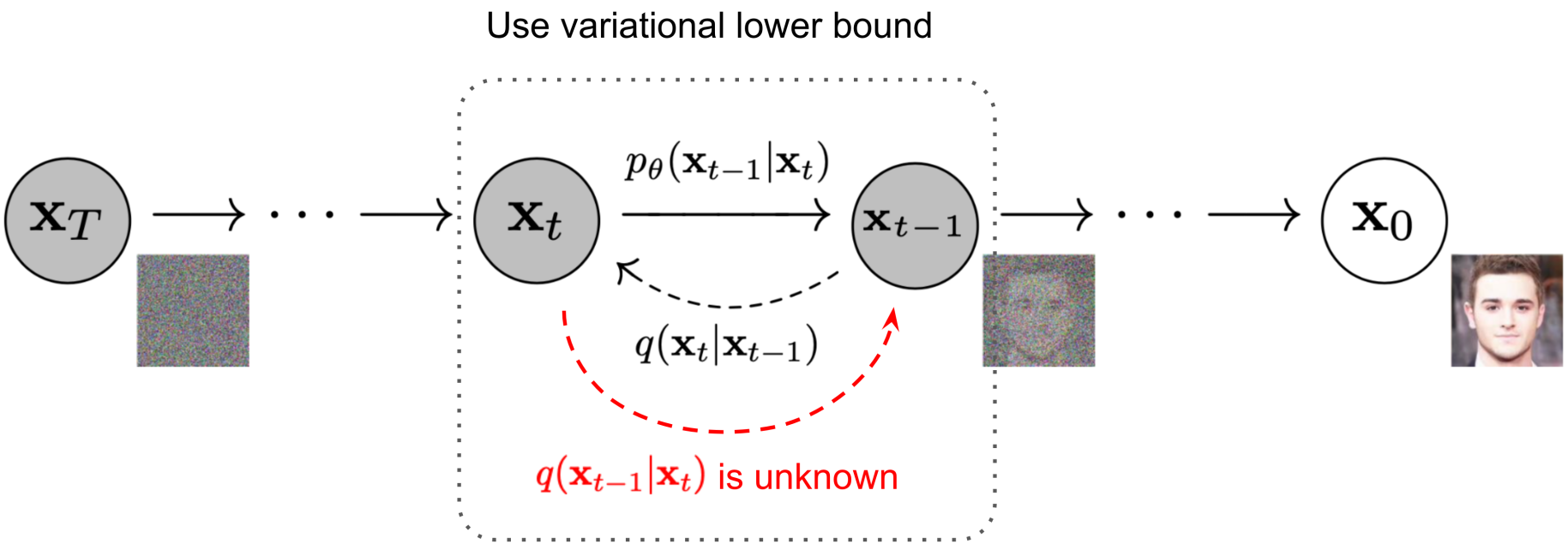
Reverse Denoising Process

- **Learn reverse Markov process:**

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \sigma_t^2 I)$$

- **Complete reverse(generative) process:**

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



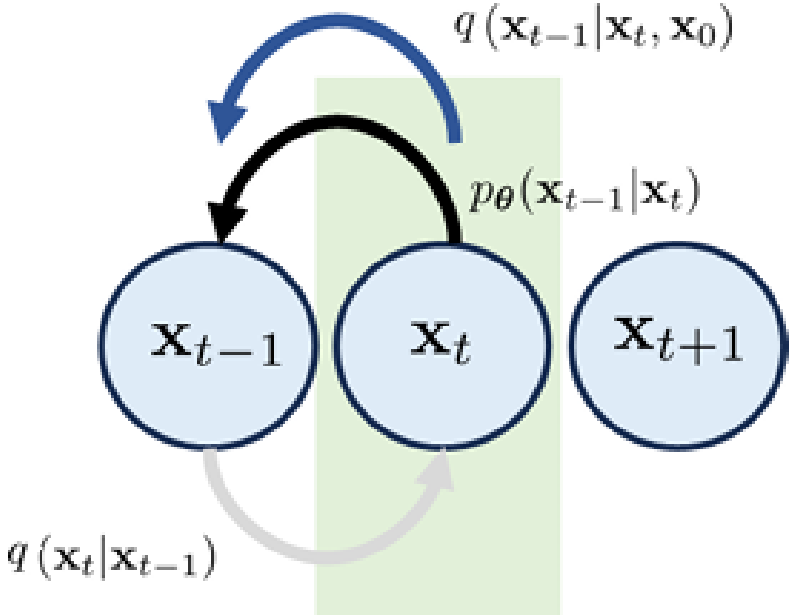
Variational Lower Bound

ELBO for Variational Diffusion Model

Let $\mathbf{x} = \mathbf{x}_0$, and $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$. The ELBO for a variational diffusion model can be written as:

$$\text{ELBO}_\theta(\mathbf{x}) = \underbrace{\mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)}\left[\log p_\theta(\mathbf{x}_0 \mid \mathbf{x}_1)\right]}_{\text{reconstruction term}} - \underbrace{\mathbb{D}_{\text{KL}}\left(q(\mathbf{x}_T \mid \mathbf{x}_0) \| p(\mathbf{x}_T)\right)}_{\text{prior matching}}$$
$$- \underbrace{\sum_{t=2}^{T} \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}\left[\mathbb{D}_{\text{KL}}\left(q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) \| p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)\right)\right]}_{\text{consistency}}.$$

Objective function



- **Why maximize ELBO?**

$$\log p_\theta(\mathbf{x_0}) \geq \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)}\right] = ELBO_\theta(x_0) \quad \text{(Using Jensen's inequality)}$$

Objective function

- Note that, $q(x_{t-1}|x_t, x_0)$ is still a Gaussian.
- Since it is a Gaussian, it is fully characterized by the mean and covariance. It turns out that,

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0)q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)}$$
$$\propto \mathcal{N}(\mathbf{x}_{t-1} \mid \boldsymbol{\mu}_q(\mathbf{x}_t, \mathbf{x}_0), \boldsymbol{\Sigma}_q(t))$$

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

$$\textbf{and } \boldsymbol{\Sigma}_q(t) = \frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{I} \stackrel{\text{def}}{=} \sigma_q^2(t)\mathbf{I},$$

where $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$

Minimizing the loss

So, to minimize the loss we have to minimize the KL term in the summation. To compute the KL divergence, we need to do something about $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$.

The big idea here is that $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)$ is Gaussian. If we want to quickly calculate the KL divergence, then it would be good if $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ is also a Gaussian.* So, we choose $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ to be a Gaussian. Moreover, we should match the form of the mean and variance! Therefore, we define:

$$p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}\left(\mathbf{x}_{t-1} \mid \underbrace{\boldsymbol{\mu}_\theta(\mathbf{x}_t)}_{\text{neural network}}, \underbrace{\sigma_q^2(t)\mathbf{I}}_{\text{known}}\right)$$

where, we assume that the mean vector can be determined using a neural network. As for the variance, we choose the variance to be $\sigma_q^2(t)$ which is identical to that of $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)$.

- But **why do we need to learn** the reverse distribution $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ if we already knew $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)$? (Think!)

Minimizing the loss

Therefore, the KL divergence is simplified to:

$$\mathbb{D}_{\mathrm{KL}}\left(q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) \,\|\, p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)\right)$$

$$= \mathbb{D}_{\mathrm{KL}}\left(\mathcal{N}(\mathbf{x}_{t-1} \mid \boldsymbol{\mu}_q(\mathbf{x}_t, \mathbf{x}_0), \sigma_q^2(t)\mathbf{I}) \,\|\, \mathcal{N}(\mathbf{x}_{t-1} \mid \boldsymbol{\mu}_\theta(\mathbf{x}_t), \sigma_q^2(t)\mathbf{I})\right)$$

$$= \frac{1}{2\sigma_q^2(t)} \left\|\boldsymbol{\mu}_q(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_t)\right\|^2$$

where, we use the fact that the KL divergence between two identical variance Gaussians is just the Euclidean distance squared between the two mean vectors multiplied by some contant.

- So, essentially our problem of optimizing a diffusion model boils down to predicting the mean of an arbitrary noisified image at an arbitrary timestep.

Minimizing the loss

In other words, we want to optimize a $\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t)$ at any time t, that matches $\mu_q(\boldsymbol{x}_t, \boldsymbol{x}_0)$ which takes the form:

$$\mu_q(\boldsymbol{x}_t, \boldsymbol{x}_0) = \frac{\sqrt{\bar{\alpha}_t}(1-\bar{\alpha}_t)\boldsymbol{x}_t + \sqrt{1-\bar{\alpha}_t}\sqrt{1-\alpha_t}\boldsymbol{x}_0}{1-\bar{\alpha}_t}$$

As $\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$ also conditions on $\boldsymbol{x}_t$, we can match $\mu_q(\boldsymbol{x}_t, \boldsymbol{x}_0)$ closely by setting it to the following form:

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

where $\hat{\boldsymbol{x}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$ is parameterized by a neural network that seeks to predict $\boldsymbol{x}_0$ from noisy image $\boldsymbol{x}_t$ and time index t. Then, the optimization problem simplifies to:

$$\arg\min_{\boldsymbol{\theta}} D_{KL}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \,||\, p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))$$

$$\vdots$$

$$= \arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)} \frac{\bar{\alpha}_{t-1}(1-\alpha_t)}{(1-\bar{\alpha}_t)^2} \|\hat{\boldsymbol{x}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) - \boldsymbol{x}_0\|_2^2$$

Final Objective function

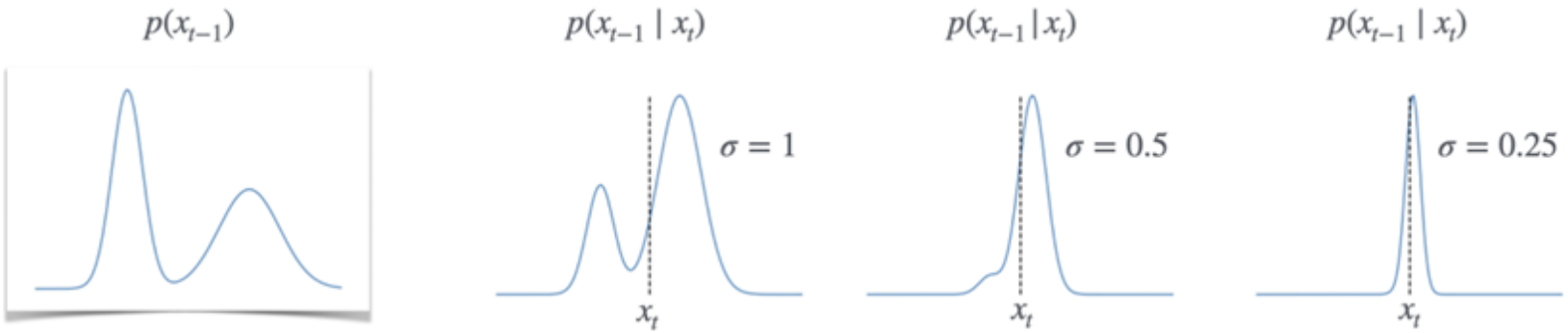
We further simplify the ELBO so that we can absorb the reconstruction term $\mathbb{E}_{q(x_1|x_0)}[\log p_\theta(x_0 \mid x_1)]$ into the KL summation term. This results into:

$$\text{ELBO}_\theta(\mathbf{x}) = -\sum_{t=1}^{T} \frac{1}{2\sigma_q^2(t)} \cdot \frac{(1-\alpha_t)^2 \bar{\alpha}_{t-1}}{(1-\bar{\alpha}_t)^2} \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} \left[\|\hat{\mathbf{x}}_\theta(\mathbf{x}_t) - \mathbf{x}_0\|^2 \right]$$

- So, Optimizing a Diffusion model again simplifies to learn a neural network to predict the original ground truth image (x_0) from an arbitrary noisified version of it (x_t). Thus, **all we need is just to train denoiser.**

Gaussian Approximation of Reverse Process

- Most of the times, the reverse sampling distribution $p_\theta(x_{t-1} \mid x_t)$ can be complex to learn and it is **not necessarily always a Gaussian**.
- However, when the noise level σ is small, the reverse distribution simplifies significantly due to the fact that: **For small σ, the reverse distribution becomes approximately Gaussian**. That is, for all times t and conditioning x_t there exists some mean parameter $\mu_t \in \mathbb{R}^d$ s.t. $p_\theta(x_{t-1} \mid x_t) \approx \mathcal{N}(x_{t-1} \mid \mu_t, \sigma^2 \mathbf{I})$.



- Moreover, this approximation holds for all time steps t and all conditions $x_t \in \mathbb{R}^d$.

Training DDPM

The forward diffusion does not require any training. If we have a clean image sample $x_0^{(m)}$, we can run the forward diffusion and prepare the images $x_1^{(m)}, ..., x_T^{(m)}$ directly.

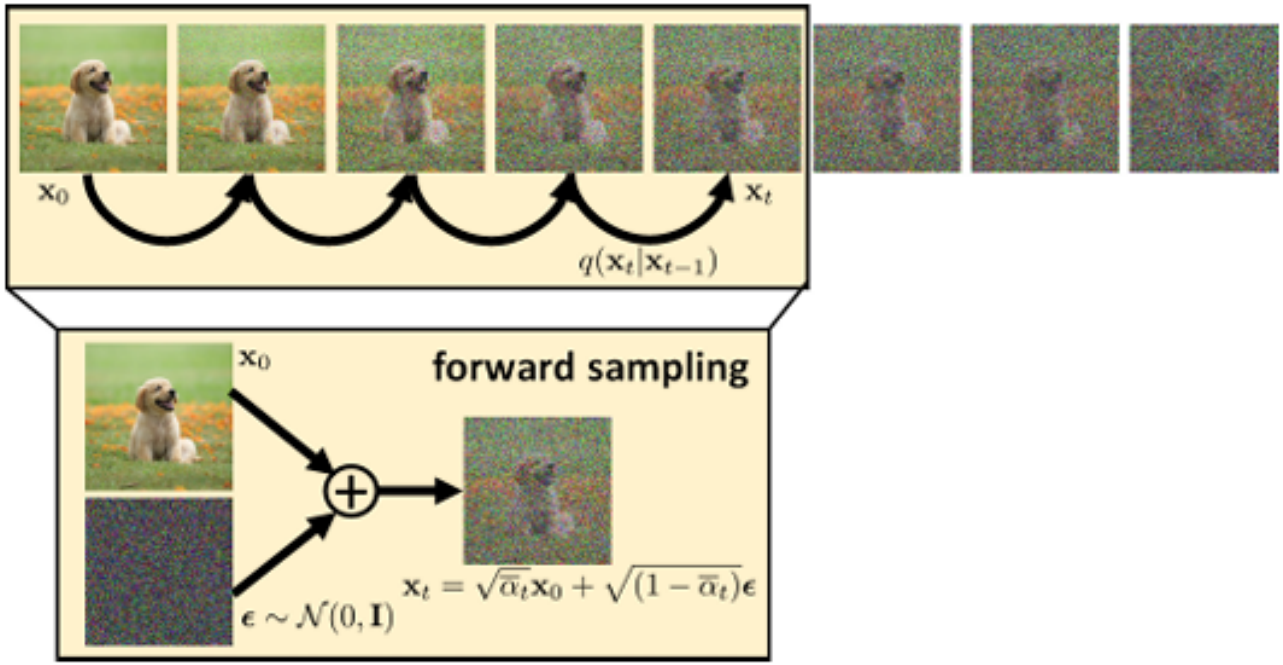


Figure: Forward diffusion process

Training DDPM

Once the training samples $x_0^{(m)}, x_1^{(m)}, \ldots, x_T^{(m)}$ are prepared, we can train the DDPM for the reverse process.

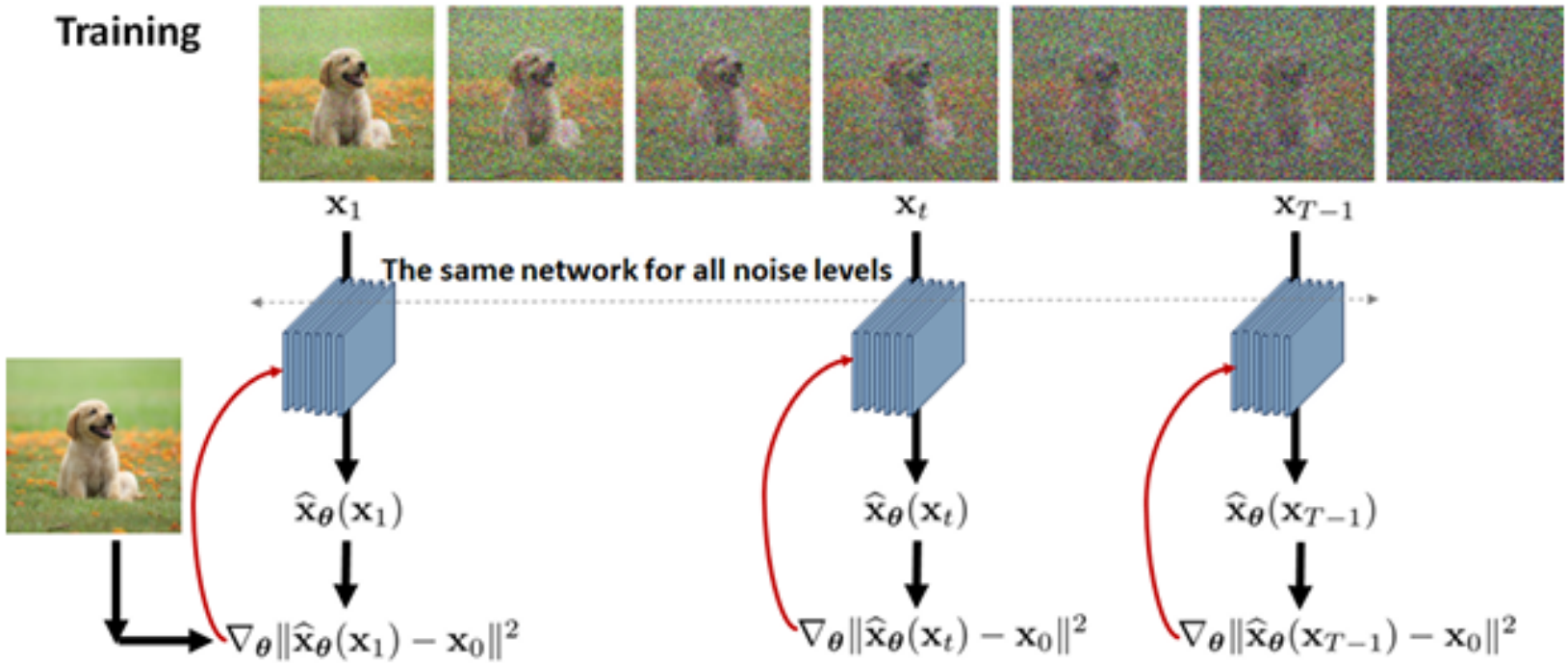


Figure: Training of a DDPM

Training DDPM

Algorithm 1	**Training of DDPM**
1:	**repeat**
2:	$x_0 \sim q(x_0)$
3:	$t \sim \text{Uniform}(\{1, \ldots, T\})$
4:	$\epsilon \sim \mathcal{N}(0, \mathbf{I})$
5:	Take gradient descent step on $\nabla_\theta \left\| x_0 - x_\theta \left(\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\, \epsilon, t \right) \right\|^2$
6:	**until** converged

Sampling from our model

Once the denoiser $\hat{x}_\theta$ is trained, we can apply it to do the inference. The inference is about sampling images from the distribution $p_\theta(x_{t-1} \mid x_t)$ over the seqeuence $x_T, x_{T-1}, ..., x_1$.

- We have to do it recursively via:
  $x_{t-1} \sim p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}|\boldsymbol{\mu}_\theta(\mathbf{x}_t), \sigma_q^2(t)\mathbf{I})$.

- By reparameterization, we have:

$$\mathbf{x}_{t-1} = \boldsymbol{\mu}_\theta(\mathbf{x}_t) + \sigma_q(t)\boldsymbol{\epsilon}, \quad \text{where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

  And this can be further expanded as:

$$\mathbf{x}_{t-1} = \frac{(1-\bar{\alpha}_{t-1})\sqrt{\alpha_t}}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{(1-\alpha_t)\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_t}\widehat{\mathbf{x}}_\theta(\mathbf{x}_t) + \sigma_q(t)\boldsymbol{\epsilon}.$$

Sampling from our model

Algorithm 2	**Sampling from DDPM**
1:	$\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$
2:	**for** $t = T, \ldots, 1$ **do**
3:	$\quad \epsilon \sim \mathcal{N}(0, \mathbf{I})$ if $t > 1$, else $\epsilon = 0$
4:	$\quad \mathbf{x}_{t-1} = \frac{(1-\bar{\alpha}_{t-1})\sqrt{\alpha_t}}{1-\bar{\alpha}_t} \mathbf{x}_t + \frac{(1-\alpha_t)\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_t} \hat{\mathbf{x}}_\theta(\mathbf{x}_t) + \sigma_q(t)\epsilon$
5:	**end for**
6:	**return** $\mathbf{x}_0$

Other equivalent formulations

- **Predicting Noise:**
  In the denoising liturature instead of predicting actual image, what we actually do is that, we use the residue-type of algorithm that predicts the noise instead of the signal. The same spirit applies denoising diffusion, where we can learn to predict the noise.

  From forward noising process via Reparameterization trick, we have the relation:

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}_0 \quad \Rightarrow \quad \mathbf{x}_0 = \frac{\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}_0}{\sqrt{\bar{\alpha}_t}}$$

  Substitute into $\boldsymbol{\mu}_q(\mathbf{x}_t, \mathbf{x}_0)$:

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

$$\vdots$$

$$= \frac{1}{\sqrt{\alpha_t}}\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\boldsymbol{\epsilon}_0$$

Noise Prediction Objective

Evaluate the model's mean via noise estimator:

$$\boldsymbol{\mu}_\theta(\mathbf{x}_t) = \frac{1}{\sqrt{\alpha_t}}\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\,\widehat{\boldsymbol{\epsilon}}_\theta(\mathbf{x}_t)$$

Final ELBO expression:

$$\text{ELBO}_\theta(\mathbf{x}_0, \boldsymbol{\epsilon}_0) = -\sum_{t=1}^{T} \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} \left[\frac{1}{2\sigma_q^2(t)} \left\| \frac{1-\alpha_t}{\sqrt{(1-\bar{\alpha}_t)}\sqrt{\alpha_t}} \left(\boldsymbol{\epsilon}_0 - \widehat{\boldsymbol{\epsilon}}_\theta(\mathbf{x}_t)\right) \right\|^2 \right]$$

- We have therefore shown that **learning a diffusion model by predicting the original image x_0 is equivalent to learning to predict the noise**; empirically, however, someworks have found that predicting the noise resulted in better performance.

Score based interpretation

Score-based generative models are alternative approaches to generate data from a desired distribution where samples are produced via **Langevin dynamics** using gradients of the data distribution (**Stein's Score function**) estimated with some score matching technique.

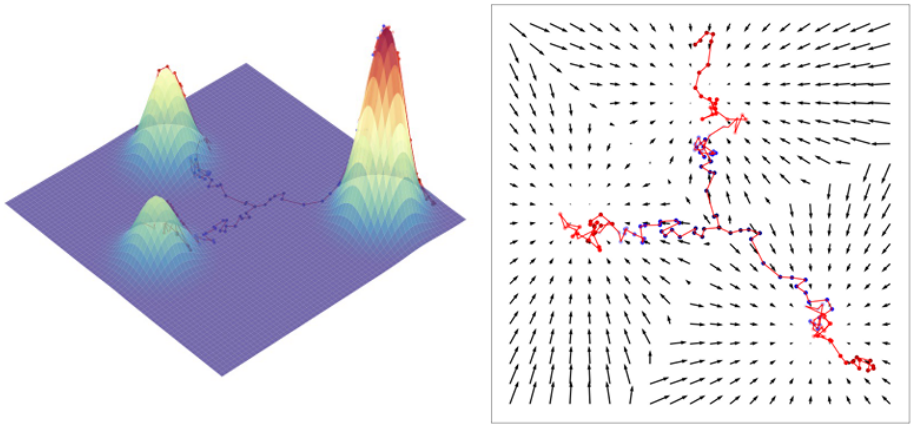


Figure: Score field

- The connection between **Variational Diffusion Models (VDMs)** and **Score-Based Generative Models (SGMs)** emerges from their shared mathematical foundations via Tweedie's formula.

Connecting Diffusion Models with Score-Based Methods

- Tweedie's Formula states that the **"true mean"** of an exponential family distribution, given samples drawn from it, can be estimated by the maximum likelihood estimate of the samples (which is the **"empirical mean"**) **"plus some correction term"** involving the score of the estimate.
- For noisy observation $\boldsymbol{x}_t \sim \mathcal{N}(\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$, Tweedie's Formula gives:

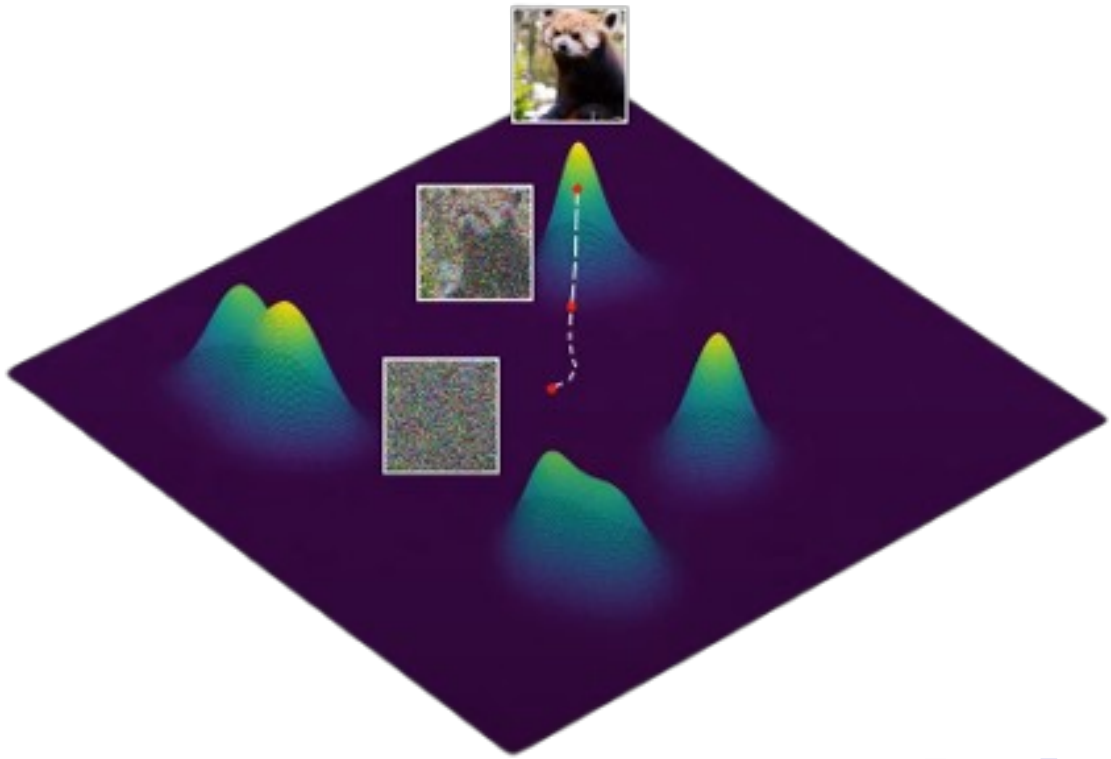
$$\sqrt{\bar{\alpha}_t}\mathbb{E}[\boldsymbol{x}_0 \mid \boldsymbol{x}_t] = \boldsymbol{x}_t + (1-\bar{\alpha}_t)\nabla \log p(\boldsymbol{x}_t) \quad \Rightarrow \quad \nabla \log p(\boldsymbol{x}_t) = \frac{\sqrt{\bar{\alpha}_t}\mathbb{E}[\boldsymbol{x}_0 \mid \boldsymbol{x}_t] - \boldsymbol{x}_t}{1-\bar{\alpha}_t}$$

- **Interpretation**:
  - **VDMs**: Predict $\boldsymbol{x}_0$, $\boldsymbol{\epsilon}_0$
  - **SGMs**: Learn score function directly

Bridge: Score prediction $\Leftrightarrow$ learning a denoiser (Thanks to Tweedie!)

Score-based Generative Models

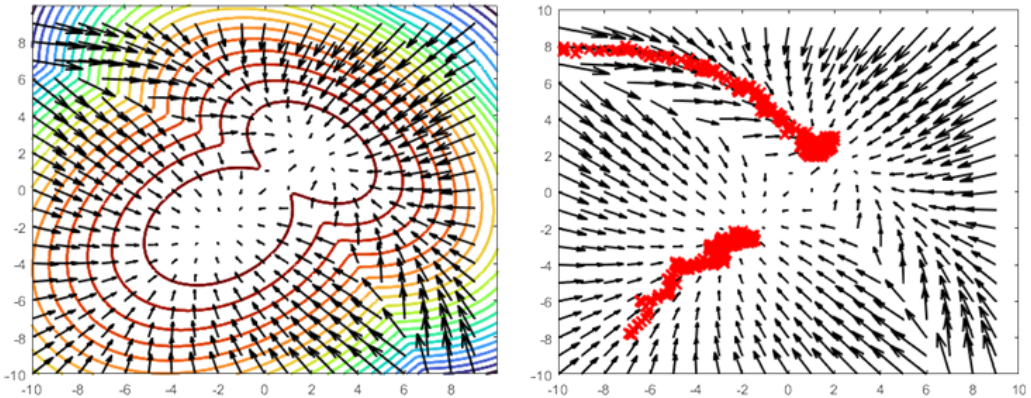
- Score based generative models learn **the score function $\nabla \log p(\boldsymbol{x})$, which points toward high-density regions in data space**. The magnitude of the vectors are the strongest at places where the change of $\log p(\boldsymbol{x})$ is maximum. Therefore, in regions where $\log p(\boldsymbol{x})$ is close to the peak will be mostly very weak gradient.



Langevin Dynamics

- For sampling we use Langevin dynamics:

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \tau \nabla \log p(\boldsymbol{x}_t) + \sqrt{2\tau}\boldsymbol{\epsilon}$$



(a) vector field of $\nabla_{\mathbf{x}} \log p(\mathbf{x})$ (b) $\mathbf{x}_t$ trajectory

- If we consider a data point living the space, the Langevin dynamics equation will basically **move the data points along the direction pointed by the vector field towards the mode**.
- $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is an extra noise term to ensure that the generated samples do not always collapse onto a mode, but hover around it for diversity.

Challenges with Vanilla Score Matching

But there are several issues with this technique.

1. **Low-dimensional Data Manifold**: Score undefined in the ambiebnt space.
2. **Low-density regions**: Hard to estimate scores with few samples.
3. **Mixing issues**: Score function ignores mixing weights in multimodal distributions which creates problem when different modes have approximately disjoint supports.

Tackling challenges: It turns out that these three drawbacks can be simultaneously addressed by adding multiple levels of Gaussian noise $\{\sigma_t\}_{t=1}^{L}$ to perturb the data distribution progressively.

1. This ensures the distribution does not collapse to a low-dimensional manifold becasue support of Gaussian is the whole domain.
2. "Large noise levels" improve score estimation in low-density regions.
3. Adding multiple levels of Gaussian noise with increasing variance will result in intermediate distributions that respect the ground truth mixing coefficients by smoothing out sharp boundaries of individual modes.

Solution: Multiple Noise Levels

- **Idea:** Add Gaussian noise at multiple levels $\sigma_1 > \sigma_2 > \cdots > \sigma_L$ to progressively perturb the data distribution. The resulting perturbed distribution is:

$$\begin{aligned} p_{\sigma_t}(\boldsymbol{x}_t) &= \int p(\boldsymbol{x})\, \mathcal{N}(\boldsymbol{x}_t \mid \boldsymbol{x}, \sigma_t^2 \mathbf{I})\, d\boldsymbol{x} \\ &= \left(p * \mathcal{N}(0, \sigma_t^2 \mathbf{I})\right)(\boldsymbol{x}_t) \end{aligned}$$

  where $*$ denotes convolution. This smooths the original data distribution by spreading probability mass outward in all directions.

- To generate an image, we use the **Annealed Langevin Dynamics (ALD)** to iteratively draw samples by denoising the image, where we initially use scores corresponding to the highest noise level, and gradually anneal down the noise level until it is small enough to be indistinguishable from the original data distribution.

Denoising Score Matching

- **Training Objective:** Train a single score network $\boldsymbol{s}_\theta(\boldsymbol{x}_t, t)$ to estimate the score $\nabla \log p_{\sigma_t}(\boldsymbol{x}_t)$ at each noise level:

$$\begin{aligned}\mathcal{L}_\theta &= \mathbb{E}_{p_{\sigma_t}(\boldsymbol{x}_t)}\left[\left\|\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) - \nabla \log p_{\sigma_t}(\boldsymbol{x}_t)\right\|^2\right] \\ &= \mathbb{E}_{p_{\sigma_t}(\boldsymbol{x}_t)}\left[\left\|\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) - \nabla \log p_{\sigma_t}(\boldsymbol{x}_t \mid \boldsymbol{x}^{(m)})\right\|^2\right] + C\end{aligned}$$

  where $\boldsymbol{x}^{(m)} \sim p(\boldsymbol{x})$ is a clean data sample available during training.

- **Why "Denoising"?** The perturbed sample $\boldsymbol{x}_t \sim \mathcal{N}(\boldsymbol{x}_t; \boldsymbol{x}^{(m)}, \sigma_t^2 \mathbf{I})$ is a noisy version of $\boldsymbol{x}^{(m)}$. The optimal score function $\nabla \log p_{\sigma_t}(\boldsymbol{x}_t \mid \boldsymbol{x}^{(m)})$ points back to the clean sample, effectively learning to "denoise" $\boldsymbol{x}_t$. Hence, minimizing this loss teaches the model to estimate a direction from noise toward the data manifold—thus the name **denoising score matching**.

Annealed Langevin Dynamics: Intuition

We start **Annealed Langevin Dynamics** by initializing samples from a fixed prior distribution, such as uniform noise. Then we:

- Run Langevin dynamics to sample from $p_{\sigma_1}(\mathbf{x})$ using a step size α_1.
- Use the final samples from this run as initialization for sampling from $p_{\sigma_2}(\mathbf{x})$, using a smaller step size α_2.
- Continue this process sequentially: each level σ_i uses the output of σ_{i-1} as its input.
- The step size is annealed using the schedule: $\alpha_i = \epsilon \cdot \frac{\sigma_i^2}{\sigma_L^2}$,where ϵ is a constant and σ_L is the smallest noise level.
- Finally, we run Langevin dynamics for $p_{\sigma_L}(\mathbf{x})$, which closely approximates the true data distribution $p_{\text{data}}(\mathbf{x})$ when $\sigma_L \approx 0$.

Sampling (via ALD)

Algorithm	Annealed Langevin Dynamics
1:	**Require:** $\{\sigma_i\}_{i=1}^{L}$, ϵ, T
2:	Initialize $\tilde{\mathbf{x}}_0$
3:	**for** $i = 1$ **to** L **do**
4:	$\quad \alpha_i \leftarrow \epsilon \cdot \sigma_i^2 / \sigma_L^2$
5:	$\quad$ **for** $t = 1$ **to** T **do**
6:	$\quad\quad$ Draw $z_t \sim \mathcal{N}(0, \mathbf{I})$
7:	$\quad\quad \tilde{\mathbf{x}}_t \leftarrow \tilde{\mathbf{x}}_{t-1} + \frac{\alpha_i}{2} s_\theta(\tilde{\mathbf{x}}_{t-1}, \sigma_i) + \sqrt{\alpha_i} z_t$
8:	$\quad$ **end for**
9:	**end for**
10:	$\mathbf{x}_0 \leftarrow \tilde{\mathbf{x}}_T$
11:	**return** $\tilde{\mathbf{x}}_T$

Diffusion as SDEs

- So far, we have seen that using only a few steps in a diffusion model results in poor performance. As the number of steps increases, the model tends to perform better. What if we use a huge number of steps? What if we take the limit as the step size $\Delta t \to 0$? Then we are left with a **continuous diffusion process**! In the continuous limit, as $\Delta t \to 0$, our discrete diffusion process turns into a **Stochastic Differential Equation (SDE)**.
- **General Itô SDE (forward process):**

$$dx = f(x,t)\,dt + g(t)\,dw, \quad w \text{ is standard Brownian motion}$$

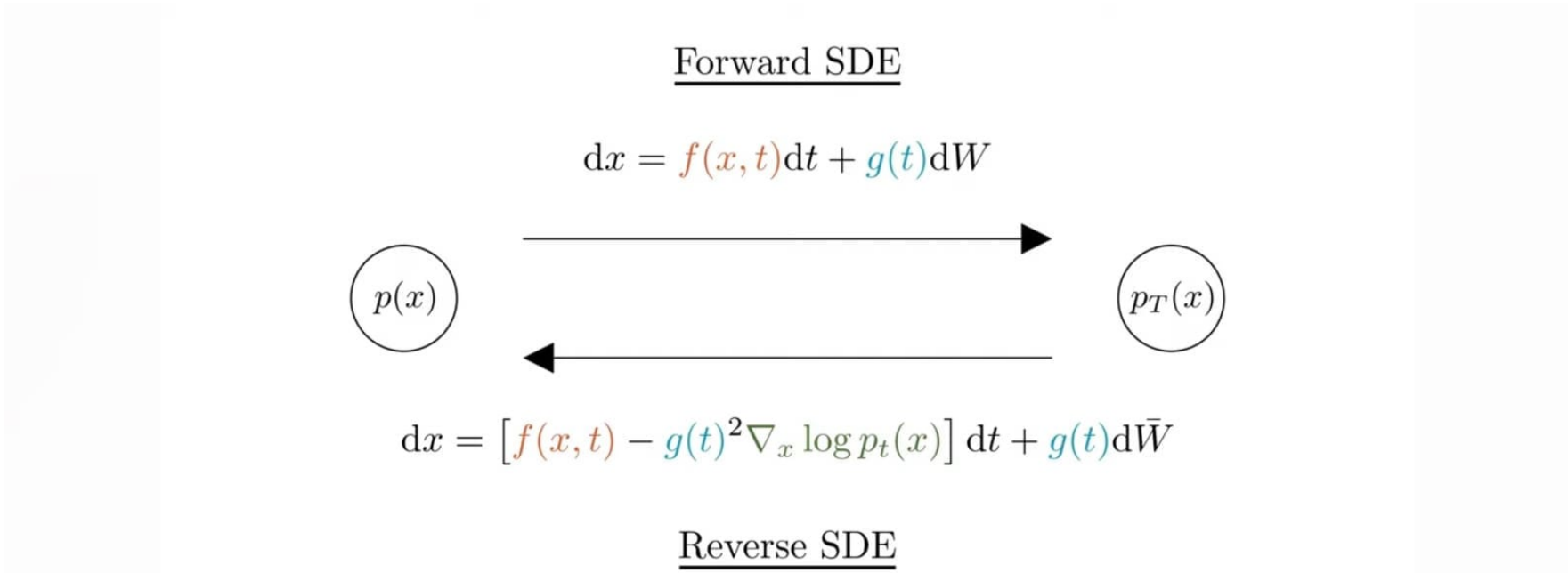
- This describes how data is gradually corrupted by noise over time:
  - $f(x,t)$ is the **drift term**, which encodes deterministic trends in the dynamics.
  - $g(t)$ is the **diffusion coefficient**, controlling the amount of random noise added.
  - dw is an infinitesimal increment of Brownian motion, representing stochasticity.

Reverse Time SDE

- But we want the reverse of this stochastic path? The time-reversal of an SDE runs the process backward in time. Reverse time SDEs are the continuous-time analog of samplers like DDPM.
- **Reverse-time SDE** (Anderson, 1982):

$$dx = \left[f(x,t) - g(t)^2 \nabla_x \log p_t(x)\right] dt + g(t)\, d\bar{w}$$

- This formulation requires estimation of the score function $\nabla_x \log p_t(x)$



Connection to DDPM

- **DDPM forward step:**

$$x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\,\epsilon, \quad \epsilon \sim \mathcal{N}(0, I)$$

- **Forward SDE (Variance Preserving):**

$$dx = -\frac{1}{2}\beta(t)\,x\,dt + \sqrt{\beta(t)}\,dw$$

- **Reverse SDE:**

$$dx = -\beta(t)\left[\frac{1}{2}x + \nabla_x \log p_t(x)\right]dt + \sqrt{\beta(t)}\,d\bar{w}$$

- DDPM is a discrete approximation of this SDE.

Connection to SMLD

- **SMLD forward step:**

$$x_i = x_{i-1} + \sqrt{\sigma_i^2 - \sigma_{i-1}^2} \cdot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I)$$

- **Forward SDE (Variance Exploding):**

$$dx = \sqrt{\frac{d[\sigma^2(t)]}{dt}}\, dw$$

- **Reverse SDE:**

$$dx = -\frac{d[\sigma^2(t)]}{dt} \nabla_x \log p_t(x)\, dt + \sqrt{\frac{d[\sigma^2(t)]}{dt}}\, d\bar{w}$$

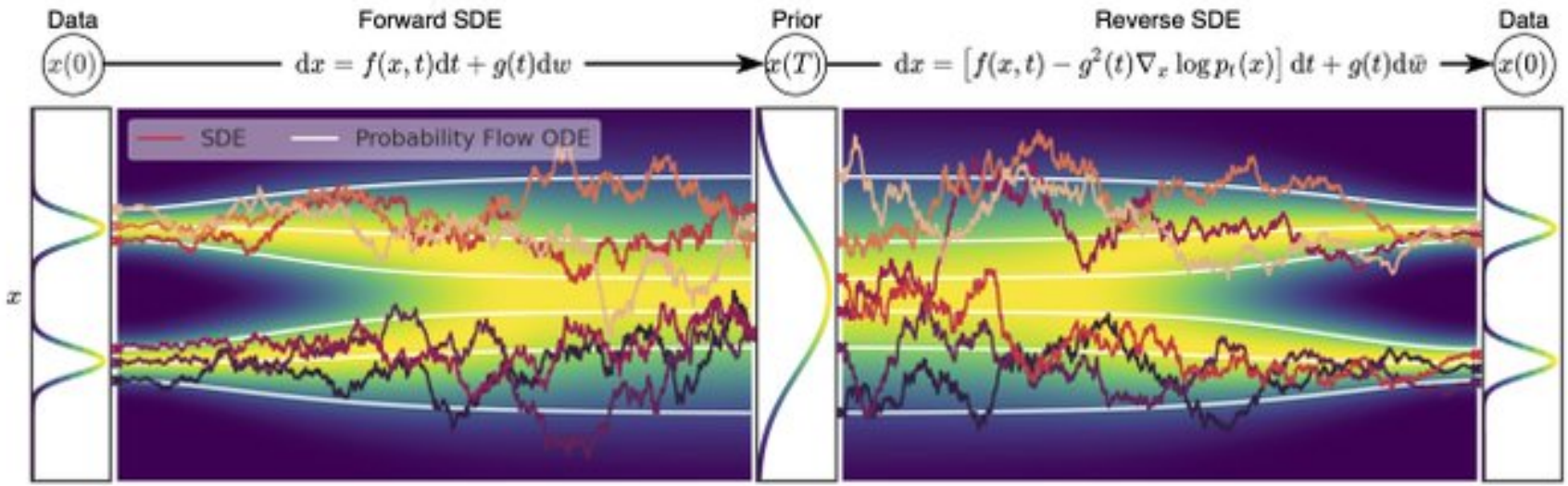
- Used in score-matching with Langevin dynamics.

Variance Preserving vs Variance Exploding

- **Variance Preserving (VP) Process:**
  - Commonly used in DDPMs (Denoising Diffusion Probabilistic Models).
  - Maintains bounded total variance: $\text{Var}(x_t) \leq 1$.
  - Easier to train due to the signal staying within a fixed dynamic range.
- **Variance Exploding (VE) Process:**
  - Commonly used in Score-based models.
  - Variance increases over time, i.e., $\text{Var}(x_t) \to \infty$ as $t \to T$.
  - Allows strong gradients at early times, improving score estimation in high-noise regions.
- **Equivalence of Inference:** As we have just seen, DDPM and SMLD correspond to two variants of the stochastic differential equation: the variance exploding (VE) and the variance preserving (VP). Despite differences in their forward processes, both produce equivalent inference (sampling) behavior. Therefore, for downstream tasks like image restoration, either VP or VE can be used without affecting final performance — though training dynamics and hyperparameter sensitivities may differ.

Probability Flow ODE

- Every SDE has an associated equivalent ODE, called **probability flow ODE**, which yields deterministic processes that sample from the same distribution as the SDE at each timestep. This establishes an equivalence to neural ODEs, allowing sampling via ODE solvers and exact computation of log-likelihoods.
- The same marginal distribution as the forward SDE can be reproduced by an ODE :
  $dx = \left[f(x,t) - \frac{1}{2}g(t)^2 \nabla_x \log p_t(x)\right] dt$, this is known as the **probability flow ODE**.
- Used in DDIM and advanced DPM-Solver methods.



Why do we need all this horrible math!?

We already have DDPM, why do we need SDE?

- SDEs can also **represent many other diffusion variants** (corresponding to different drift and diffusion terms), offering flexibility in design choices, like scaling and noise-scheduling.
- The SDE perspective is powerful because existing theory provides a general closed-form solution for the time-reversed SDE. **Discretization of the reverse-time SDE** for our particular diffusion immediately yields the sampler we derived in this section, but **reverse-time SDEs for other diffusion variants are also available** automatically (and can then be solved with any off-the-shelf or custom SDE solver), enabling better training and sampling strategies.

Types and Variants

- **Deterministic sampling with DDIM** in much lesser iterations (By allowing non-markovian forward process to skip iterations in the reverse process)
- **Knowledge Distillation (for e.g. Score Distillation)** for fast inference (Progressively distill knowledge from teacher model to student model)
- **Consistency Trained Models(CTM)** for single step generation (learns to map any intermediate noisy data point x_t on the diffusion sampling trajectory back to its origin x_0 directly)
- **Predictor-Corrector models** via faster ODE/SDE solvers by correcting our predicted path in each iteration
- **Cascaded diffusion** for improved quality via increasing resolution (Refine details at each stage by Up-sampling our model)
- **Latent Diffusion Models(LDM)** for reducing the cost (Operate in VAE latent space, not pixel space)

Strengths and Limitations

Strengths

- **Stable Training:** Avoids mode collapse and adversarial instability seen in GANs.
- **High Sample Quality:** Generates sharp, realistic images competitive with state-of-the-art.
- **Flexible Conditioning:** Easily supports text, image, class, and semantic guidance.
- **Theoretically Grounded:** Based on score matching and variational principles.
- **Uncertainty Modeling:** Naturally captures distributional uncertainty (probabilistic).

Strengths and Limitations

Limitations

- **Slow Sampling:** Requires hundreds of steps for good results (though improving).
- **High Computational Cost:** Training is expensive and memory intensive.
- **Denoising Bias:** Errors may accumulate during long sampling chains.
- **Limited Interpretability:** Hard to understand internal representations.
- **Less Direct Control:** "Fine-grained controllability" is still an open challenge. For e.g. editing or manipulating specific parts of the output is still challenging and requires extra tricks.

Challenges and Future directions

- Traditional diffusion models often require **hundreds of sampling steps**, making inference slow.
- **Discretization Methods:**
  - **Euler-Maruyama (1st order):** Simple but less accurate.
  - **Heun's Method (2nd order):** Improved stability and precision.
  - **Runge-Kutta (RK4):** High-order method with better accuracy but higher compute.
- **Recent Developments:**
  - **DPM-Solver:** A fast ODE-based sampler achieving high-quality samples with fewer steps.
  - **Score Distillation and Continuous-Time Diffusion:** Use learned scores and SDE/ODE solvers for more efficient generation.
- **Goal:** Achieve faster and more efficient sampling without compromising generation quality.

Conclusion

- Tractability and flexibility are two conflicting objectives in generative modeling.
- **Tractable models** can be analytically evaluated and cheaply fit data (e.g. via a Gaussian), but they cannot easily describe the structure in rich datasets.
- **Flexible models** can fit arbitrary structures in data, but evaluating, training, or sampling from these models is usually expensive.

Diffusion models are both analytically tractable and flexible.

References

- Sohl-Dickstein. (2015). Deep Unsupervised Learning Using Non-equilibrium Thermodynamics.
- Lilian Weng. (2021). What are Diffusion Models? *Lil'Log*.
- Calvin Luo. (2022). Understanding Diffusion Models: A Unified Perspective.
- Ho et al. (2020). Denoising Diffusion Probabilistic Models.
- Stanley Chan. (2025). Tutorial on Diffusion Models for Imaging and Vision.
- Song - Ermon. (2020). Generative Modelling by Estimating Gradients of the Data Distribution.
- Song - Ermon. (2020). Improved Techniques for Training Score-Based Generative Models.
- Advani et al. (2024). Step-by-step Diffusion: An elementary tutorial.

Q&A

Questions?

Thank you!