Generative AI and LLM

Diffusion models

CS5202

Course Instructor : Dr. Nidhi Goyal

Recap

- Diffusion models

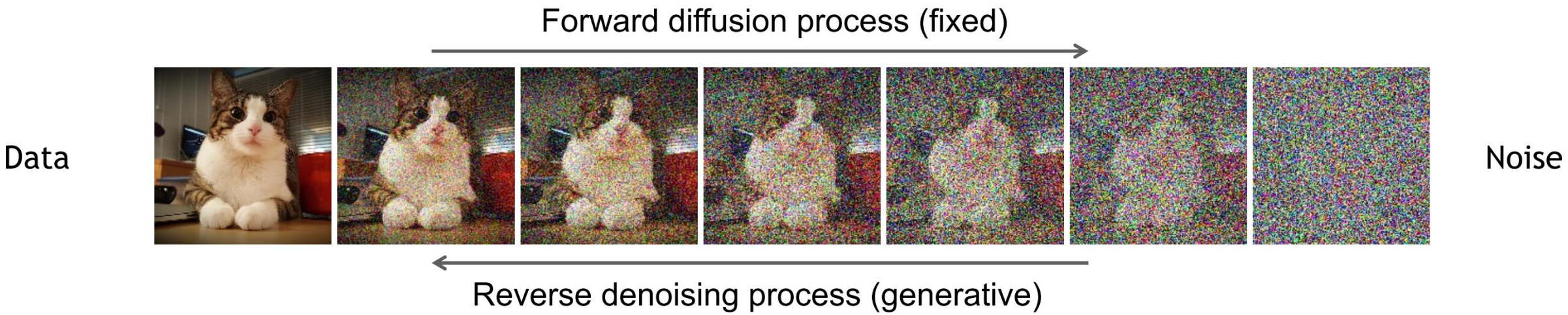
Diffusion models

- The essential idea is to destroy the structure systematically and slowly in data distribution through iterative forward diffusion process.
- We then reverse diffusion process that restores structure in data, yielding a highly flexible and tractable generative model of data.

Denoising Diffusion Models

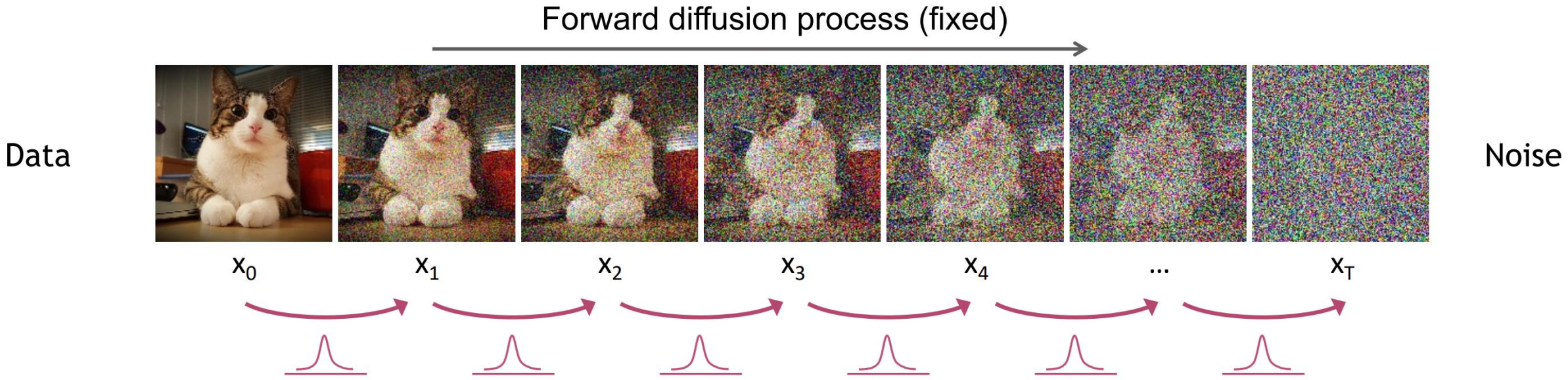
Denoising diffusion models consist of two processes:

- Forward diffusion process that gradually adds noise to input
- Reverse denoising process that learns to generate data by denoising



Forward Diffusion Process

- The formal definition of the forward process in T steps:

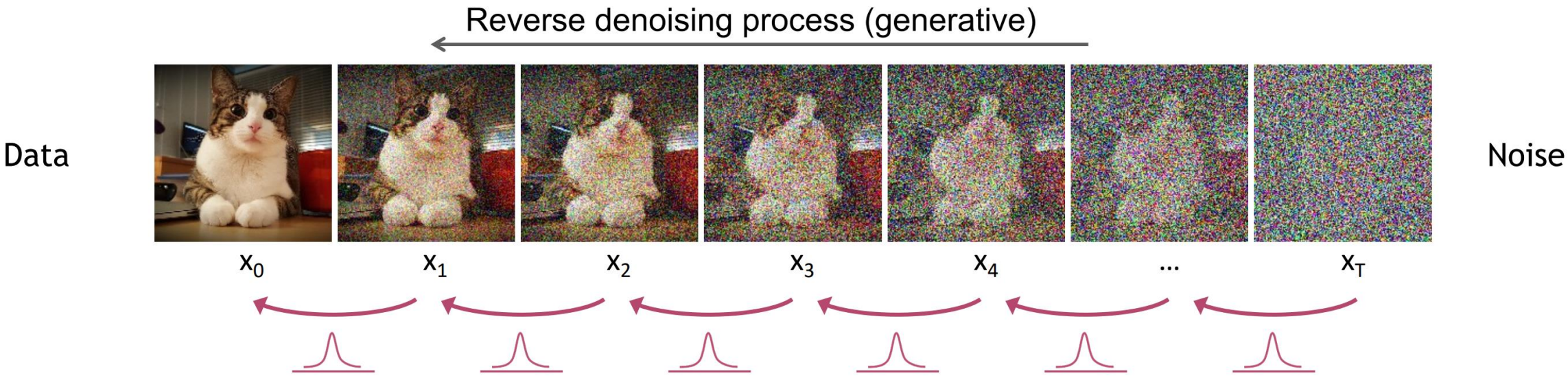


$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x_t}; \sqrt{1-\beta_t}\mathbf{x_{t-1}}, \beta_t\mathbf{I}) \quad \Rightarrow \quad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$$

(joint)

Reverse Diffusion Process

- Formal definition of forward and reverse processes in T steps:



$$p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$$
$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \underbrace{\mu_\theta(\mathbf{x}_t, t)}, \sigma_t^2\mathbf{I})$$

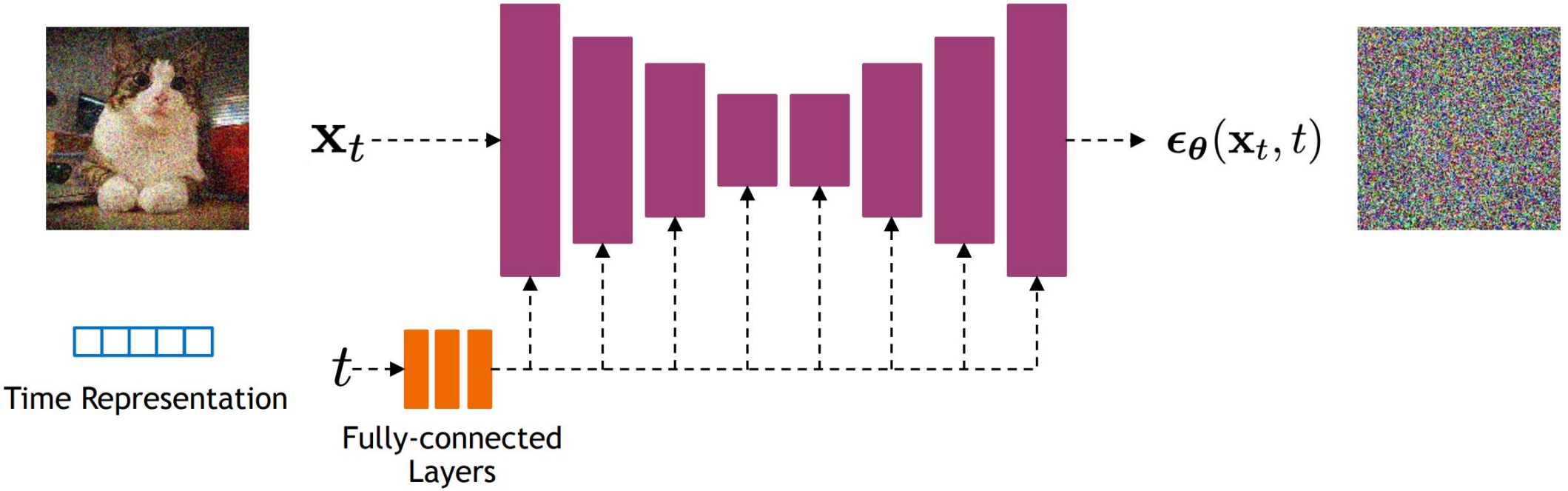
Trainable network
(U-net, Denoising Autoencoder)

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

Similar to the generative model in hierarchical VAEs.

Implementation Architectures

- Diffusion models often use U-Net architectures with ResNet blocks and self-attention layers



Intuition/Math Forward process

- Refer class notes

ELBO Recap

Why use ELBO?

Directly maximizing $p(x)$ is very difficult:

- it involves either marginalizing over the entire latent space $\boldsymbol{Z}$ (intractable for complex models) OR
- It involves having access to the ground truth latent encoder $p(z|x)$

ELBO:

$$\log\big(p(x)\big) \geq \mathbb{E}_{q_\phi(z\,|\,x)}\left[\log \frac{p(x,z)}{q_\phi(z|x)}\right]$$

Question: Why does the $\geq$ show up here? → With the derivation in the appendix, we see a $D_{KL}(q_\phi(z|x)\,||\,p(z|x))$ term show up which is always ≥ 0.

Applying chain-rule of probabilities:

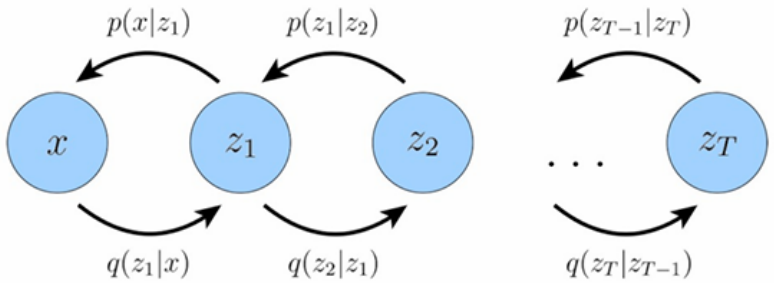
$$ELBO = \mathbb{E}_{q_\phi(z\,|\,x)}[\log p_\theta(x|z)] - D_{KL}\big(q_\phi(z\,|\,x)||\,|p(z)\big)$$

Reconstruction Prior matching

https://deeplearning.cs.cmu.edu/F23/document/slides/lec23.diffusion.pdf

Intuition/Math Reverse process

Markovian Hierarchical Variational Autoencoder



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

Posterior probability: $q_\phi(z_{1:T} \mid x) = q_\phi(z_1 \mid x) \prod_{t=2}^{T} q_\phi(z_t|z_{t-1})$

Updated ELBO:

$$\log(p(x)) \geq \mathbb{E}_{q_\phi(z_{1:T} \mid x)} \left[\log \frac{p(x, z_{1:T})}{q_\phi(z_{1:T} \mid x)} \right]$$

https://deeplearning.cs.cmu.edu/F23/document/slides/lec23.diffusion.pdf

Intuition/Math Reverse process

Diffusion models are essentially **MHVAEs** with **3 restrictions:**

1. Latent dimension is the same as the data dimension
2. The encoder has no parameters to be learnt. It is defined to be a linear gaussian such that the t^{th} gaussian is centered around the previous latent z_{t-1}
3. The parameters for the gaussians are scheduled such that the final latent is a standard gaussian.

$$z_T \sim \mathcal{N}(z_T;\ 0, \boldsymbol{I})$$

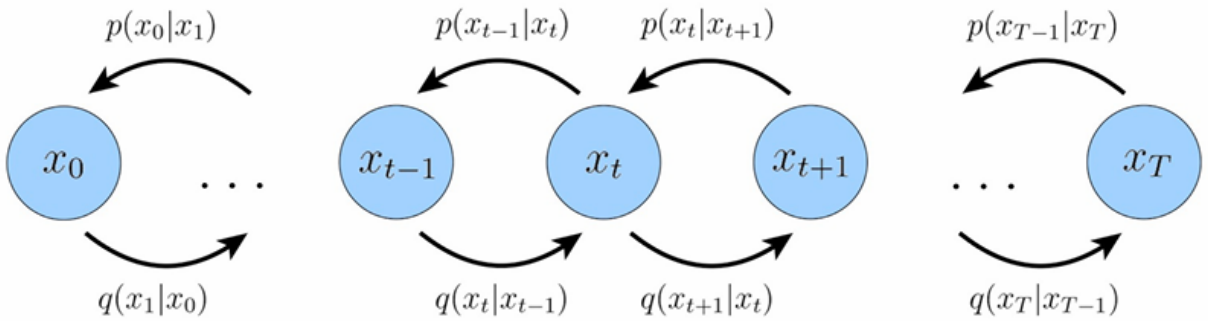
The first restriction allows for some mild abuse of notation:

$$q_\phi(x_{1:T} \mid x_0) = \prod_{t=1}^{T} q_\phi(x_t|x_{t-1})$$

$$p(x_{0:T}) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)$$

https://deeplearning.cs.cmu.edu/F23/document/slides/lec23.diffusion.pdf

Intuition/Math Reverse process



The first restriction allows for some mild abuse of notation:

$$q_\phi(x_{1:T} \mid x_0) = \prod_{t=1}^{T} q_\phi(x_t|x_{t-1})$$

$$p(x_{0:T}) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)$$

https://deeplearning.cs.cmu.edu/F23/document/slides/lec23.diffusion.pdf

Intuition/Math Reverse process

Following the second restriction, we now define the linear gaussian for the encoding (diffusion) process:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \mu_t(x_{t-1}), \Sigma_t \boldsymbol{I})$$

$$\mu_t(x_{t-1}) = \sqrt{1-\beta_t}x_{t-1}, \quad \Sigma_t = \beta_t$$

We additionally define $\alpha_t = 1 - \beta_t$.
β_t is defined to preserve variance across the diffusion steps.

We can now write

$$q(x_t|x_{t-1}) = \mathcal{N}\left(x_t; \sqrt{\alpha_t}x_{(t-1)}, (1-\alpha_t)\boldsymbol{I}\right)$$

Using the reparameterization trick:

$$x_t = \sqrt{\alpha_t}x_{(t-1)} + \left(\sqrt{1-\alpha_t}\right)\epsilon, \qquad \epsilon \sim \mathcal{N}(0, \boldsymbol{I})$$

$$= \sqrt{\alpha_t\alpha_{t-1}}x_{(t-2)} + \left(\sqrt{1-\alpha_t\alpha_{t-2}}\right)\epsilon$$

$$= \sqrt{\alpha_t\alpha_{t-1}\alpha_{t-2}}x_{(t-3)} + \left(\sqrt{1-\alpha_t\alpha_{t-2}\alpha_{t-3}}\right)\epsilon$$

$$= \sqrt{\alpha_t\alpha_{t-1}\dots\alpha_1}x_{(0)} + \left(\sqrt{1-\alpha_t\alpha_{t-2}\dots\alpha_1}\right)\epsilon$$

$$= \sqrt{\overline{\alpha_t}}\boldsymbol{x_{(0)}} + \left(\sqrt{\boldsymbol{1-\overline{\alpha_t}}}\right)\boldsymbol{\epsilon}$$

This takes us from time step 0 to t in **one step!**

From the third restriction, we get

$$\alpha_T \to 0$$

Sum of two gaussians is another gaussian with mean as the sum of the two means and variance as the sum of the two variances.

$$(1-\alpha_t)\epsilon \to \mathcal{N}(\epsilon; 0, 1-\alpha_t\boldsymbol{I})$$

Define

$$\overline{\alpha_t} = \prod_{s=1}^{t}\boldsymbol{\alpha_s}$$

https://deeplearning.cs.cmu.edu/F23/document/slides/lec23.diffusion.pdf

Generative Process

From the third assumption, we can write the exact prior on the final step x_T :

$$p(x_T) = \mathcal{N}(x_T; 0, \boldsymbol{I})$$

For all other steps, we can write a learned distribution:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_t \boldsymbol{I})$$

Exactly tractable variance

Neural Network: **U-Net**
Denoising network

$$p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)$$

https://deeplearning.cs.cmu.edu/F23/document/slides/lec23.diffusion.pdf

Updated ELBO

Diffusion Models – Updated ELBO

*Derivation in appendix!

$$\log p(x) = \log \int p(x_{0:T}) dx_{0:T}$$

$$\ldots *$$

$$= \mathbb{E}_{q(x_1 \mid x_0)}[\log p_\theta(x_0|x_1)] - \mathbb{E}_{q(x_{T-1} \mid x_0)}[D_{KL}(q(x_T \mid x_{T-1})| \,|p(x_T))]$$

$$- \sum_{t=1}^{T-1} \mathbb{E}_{q(x_{t-1}, x_{t+1} \mid x_0)}\left[D_{KL}\left(q(x_t \mid x_{t-1}) \,||p_\theta(x_t|x_{t+1})\right)\right]$$

This has **2** random variables for each **t**, this makes the computation slightly hard. We would prefer for there to be need for just **1**!

We can arbitrarily modify the diffusion process distribution to

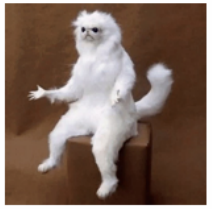
$$q(x_t \mid x_{t-1}, x_0) = \frac{q(x_{t-1} \mid x_t, x_0) q(x_t|x_0)}{q(x_{t-1} \mid x_0)}$$

https://deeplearning.cs.cmu.edu/F23/document/slides/lec23.diffusion.pdf

Updated ELBO

Diffusion Models – Updated ELBO

*Derivation in appendix!



$$\log p(x) = \log \int p(x_{0:T}) dx_{0:T}$$

$$\ldots *$$

$$= \mathbb{E}_{q(x_1 \mid x_0)}[\log p_\theta(x_0|x_1)] - D_{KL}\left(q(x_T \mid x_0)| \,|p(x_T)\right) - \sum_{t=2}^{T} \mathbb{E}_{q(x_t \mid x_0)}\left[D_{KL}\left(q(x_{t-1}| \, x_t, x_0) \,||p_\theta(x_{t-1}|x_t)\right)\right]$$

Reconstruction | Prior matching | Denoising

- Reconstruction: Reconstruction from least noisy version **(hyperparameter choice can make this arbitrarily small)**
- Prior matching: Moving the posterior closer to the true prior on the final noisy step **(0 for diffusion models)**
- Denoising: Divergence between approximate denoising (p_θ) and true denoising (q) steps

$q(x_{t-1}| \, x_t, x_0)$ is **tractable** and can be calculated **exactly** without any approximation:

$$q(x_{t-1}| \, x_t, x_0) = \mathcal{N}\left(x_{t-1}; \, \bar{\mu}_t, \Sigma_t \boldsymbol{I}\right)$$

$$\bar{\mu}_t = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})x_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)x_0}{1-\bar{\alpha}_t}, \qquad \Sigma_t = \frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}$$

https://deeplearning.cs.cmu.edu/F23/document/slides/lec23.diffusion.pdf

Loss Formulation

Diffusion Models – Loss formulation

Further, we have $x_t = \sqrt{\alpha_t} x_{(t-1)} + \left(\sqrt{1-\alpha_t}\right)\epsilon,\ \epsilon \sim \mathcal{N}(0, \boldsymbol{I})$ from definition

This lets us rewrite the true mean of the denoising process as:

$$\bar{\mu}_t = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{(1-\alpha_t)}{\sqrt{1-\bar{\alpha}_t}}\epsilon\right)$$

We can also write the predicted mean as:

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{(1-\alpha_t)}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\right)$$

This lets us reformulate the loss to present a noise prediction problem:

$$L_{t-1} = \mathbb{E}_{x_0,\epsilon}\left[\frac{(1-\alpha_t)^2}{2\Sigma_t \alpha_t (1-\bar{\alpha}_t)}\left|\left|\epsilon - \epsilon_\theta(x_t, t)\right|\right|^2\right] + C$$

https://deeplearning.cs.cmu.edu/F23/document/slides/lec23.diffusion.pdf

Diffusion models- Training and Inference

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 x_t
$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$
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** $\mathbf{x}_0$

How do we tell the model what timestep we are on?

Temporal encodings in the form or sinusoids (or anything, really)

https://deeplearning.cs.cmu.edu/F23/document/slides/lec23.diffusion.pdf

Intuition/Math Reverse process

- Diffusion models are **Markovian Hierarchical VAEs** with extra restrictions
- The loss is the vanilla VAE ELBO loss with an added denoising term
- The encoder has **0 parameters**
- The true denoising posterior can be **exactly calculated**
- The problem can be reformulated as a noise prediction problem
- There's a ton of math underlying a rather simple intuition

https://deeplearning.cs.cmu.edu/F23/document/slides/lec23.diffusion.pdf

Comparison of DDGM, VAE, and Flows

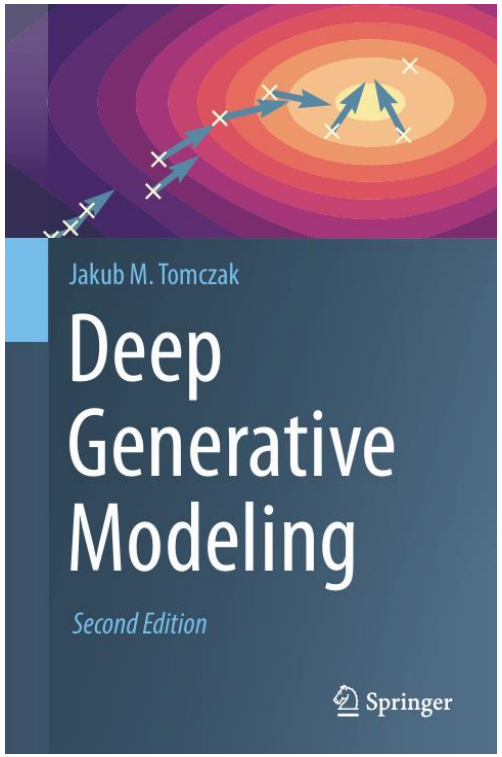
Table 5.1 A comparison among DDGMs, VAEs, and flows.

Model	Training	Likelihood	Reconstruction	Invertible	Bottleneck (latents)
DDGMs	Stable	Approximate	Difficult	No	No
VAEs	Stable	Approximate	Easy	No	Possible
Flows	Stable	Exact	Easy	Yes	No

Books and lecture notes

Deep Generative Modeling

2026



Generative AI & LLMs · Fall

References

Denoising Diffusion Probabilistic Models, https://arxiv.org/abs/2006.11239

Diffusion Models: A Comprehensive Survey of Methods and Applications, https://arxiv.org/abs/2209.00796

Understanding Diffusion Models: A Unified Perspective, https://arxiv.org/abs/2208.11970

GLOW, https://proceedings.neurips.cc/paper/2018/file/d139db6a236200b21cc7f752979132d0-Paper.pdf

Variational inference with normalizing flows, https://arxiv.org/abs/1505.05770

Normalizing Flows: An Introduction and Review of Current Methods, https://arxiv.org/abs/1908.09257

Didrik Nielsen's lecture, https://www.youtube.com/watch?v=2tVHbcUP9b8

Hans van Gorp's lecture, https://www.youtube.com/watch?v=yxVcnuRrKqQ&t=17s

Tim Salimans' lecture, https://www.youtube.com/watch?v=pea3sH6orMc