Deep Generative Models: Diffusion Models

Fall Semester 2026

René Vidal
Director of the Center for Innovation in Data Engineering and Science (IDEAS)
Rachleff University Professor, University of Pennsylvania
Amazon Scholar & Chief Scientist at NORCE

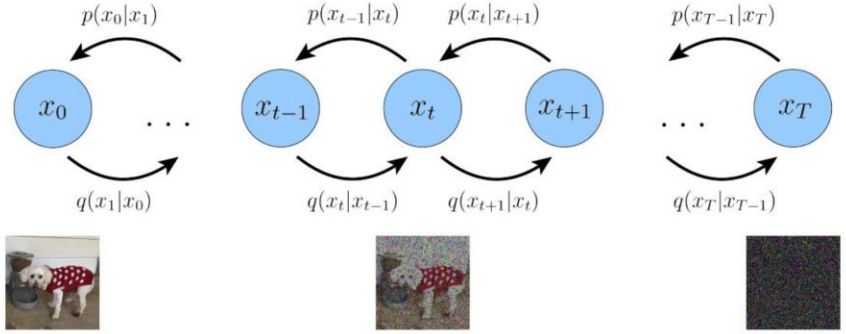


Outline

- Markov Hierarchical Variational Auto Encoders (MHVAEs)
  - Autoregressive Encoder and Autoregressive Decoder of an MHVAE
  - Derivation of the ELBO of an MHVAE
- Diffusion Models: MHVAEs with a Linear Gaussian Autoregressive Latent Space
  - Forward Diffusion Process
  - Reverse Diffusion Process
  - ELBO for Diffusion Models as a particular case of the ELBO for MHVAEs
- **Implementation Details: UNet architecture, Training and Sampling Strategies**
- Application of Diffusion Models
  - Text-Conditioned Diffusion Model: Stable Diffusion
  - Multimodal Control for Consistent Synthesis: ControlNet
  - Image Editing: DDIM, P2P

Implementation (DDPM)

- The Denoising Diffusion Probabilistic Model (DDPM) fixes the noise variances α_t of the forward process and learns only the backward (denoising) process [Ho et al., 2020].



$$q_\phi(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{\alpha_t}\boldsymbol{x}_{t-1}, (1-\alpha_t)\boldsymbol{I})$$

- For the backward process, we use reparameterization of the ELBO

$$\log p_\theta(\boldsymbol{x}_0) \geq \underbrace{\mathbb{E}_{q_\phi(\boldsymbol{x}_1|\boldsymbol{x}_0)}[\log p_\theta(\boldsymbol{x}_0 \mid \boldsymbol{x}_1)]}_{\text{reconstruction term}} - \underbrace{D_{\mathrm{KL}}\left(q_\phi(\boldsymbol{x}_T \mid \boldsymbol{x}_0) \,||\, p_\theta(\boldsymbol{x}_T)\right)}_{\text{prior matching term}} - \sum_{t=2}^{T} \underbrace{\mathbb{E}_{q_\phi(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[D_{\mathrm{KL}}\left(q_\phi(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0) \,||\, p_\theta(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)\right)\right]}_{\text{score matching term}}$$

$$= -\sum_{t=1}^{T} \underbrace{\mathbb{E}_{q_\phi(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[D_{\mathrm{KL}}\left(q_\phi(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0) \,||\, p_\theta(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)\right)\right]}_{\text{reconstruction term} + \text{score matching term}} = -\sum_{t=1}^{T} \mathbb{E}_{q_\phi(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[\frac{1}{2\sigma_q^2(t)}\frac{(1-\alpha_t)^2}{(1-\bar{\alpha}_t)\alpha_t}\|\boldsymbol{\epsilon}_0 - \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)\|_2^2\right]$$

$$\sigma_q^2(t) = \begin{cases} \sigma_{\text{rec}}^2, & t = 1 \\ \dfrac{(1-\alpha_t)(1-\overline{\alpha_{t-1}})}{1-\overline{\alpha_t}}, & t \geq 2 \end{cases}$$

- For each $\boldsymbol{x}$, we need to compute T terms in the sum, which is expensive for typical values of T (T= 1000).

SGD over time-steps

- DDPM minimizes the ELBO efficiently by performing SGD over the set of timesteps $[T] = \{1, 2, \dots, T\}$.
- Using linearity of expectation and letting $t \sim \text{Unif}([T])$

$$\text{ELBO} = -\sum_{t=1}^{T} \mathbb{E}_{q_\phi(\boldsymbol{x}_t|\boldsymbol{x}_0)} \left[\frac{1}{2\sigma_q^2(t)} \frac{(1-\alpha_t)^2}{(1-\bar{\alpha}_t)\alpha_t} \|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)\|_2^2 \right]$$

$$= -\mathbb{E}_{q_\phi(\boldsymbol{x}_t|\boldsymbol{x}_0)} \left[\sum_{t=1}^{T} \frac{1}{2\sigma_q^2(t)} \frac{(1-\alpha_t)^2}{(1-\bar{\alpha}_t)\alpha_t} \|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)\|_2^2 \right]$$

$$= -\mathbb{E}_{q_\phi(\boldsymbol{x}_t|\boldsymbol{x}_0), t} \left[\frac{T}{2\sigma_q^2(t)} \frac{(1-\alpha_t)^2}{(1-\bar{\alpha}_t)\alpha_t} \|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)\|_2^2 \right]$$

- Thus, the sampling procedure computes only 1 term instead of T terms.
- Thus, the simplified (unweighted) learning objective used in DDPM is

$$\mathbb{E}_{q_\phi(\boldsymbol{x}_t|\boldsymbol{x}_0), t} \left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)\|_2^2 \right]$$

where $\boldsymbol{x}_t = \sqrt{\overline{\alpha_t}}\boldsymbol{x}_0 + \sqrt{1 - \overline{\alpha_t}}\,\boldsymbol{\epsilon}$

Why Use a Simplified DDPM Training Objective?

- Weighted objective

$$\mathbb{E}_{q_\phi(\boldsymbol{x}_t|\boldsymbol{x}_0),t}\left[\frac{T}{2\sigma_q^2(t)}\frac{(1-\alpha_t)^2}{(1-\bar{\alpha}_t)\alpha_t}\|\boldsymbol{\epsilon}_0-\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t,t)\|_2^2\right]$$

Emphasizes certain timesteps, particularly small t under the original DDPM setup

- Unweighted objective

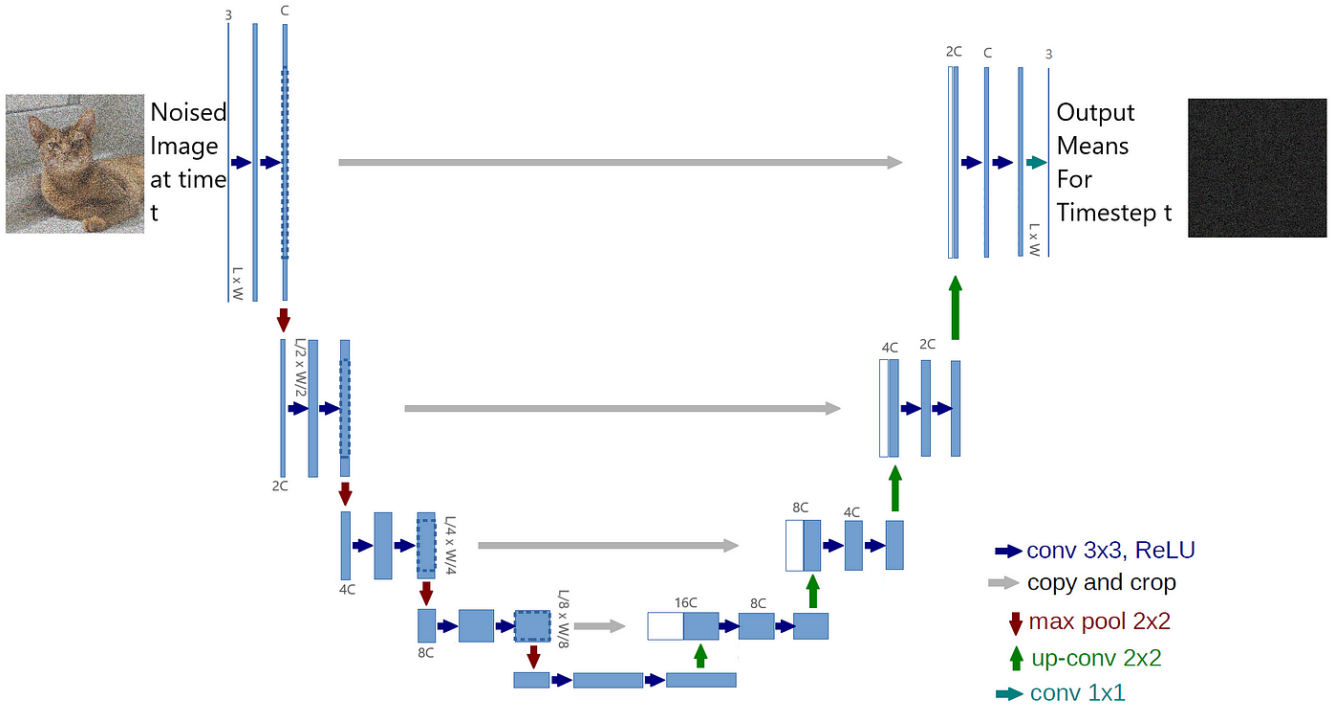
$$\mathbb{E}_{q_\phi(\boldsymbol{x}_t|\boldsymbol{x}_0),t}\left[\|\boldsymbol{\epsilon}_0-\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t,t)\|_2^2\right]$$

Removes timestep weights and gives more emphasis to challenging denoising tasks

Empirically improves generated image quality compared to weighted objective

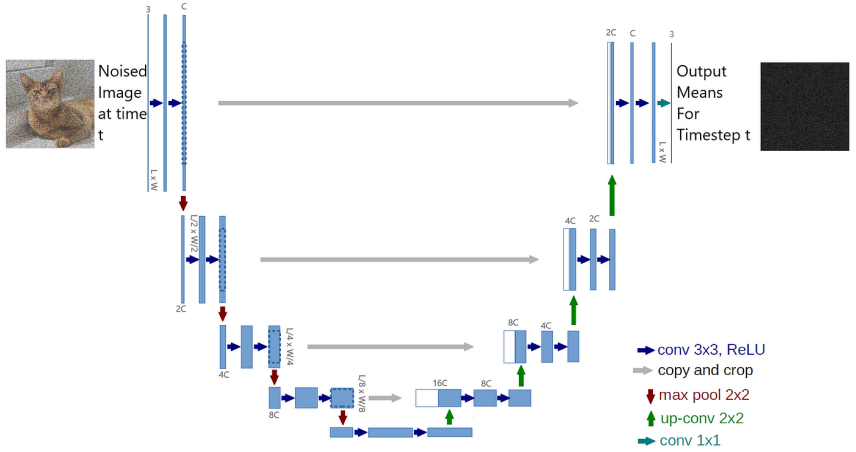
Architecture of DDPM

- DDPM uses a U-Net with residual connection and self-attention layers to represent the noise $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)$.
- Since the parameter $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)$ depends both on $\boldsymbol{x}_t$ and on time t, a time embedding needs to be provided as input to each unit of the U-Net.



Why U-Net for DDPM?

- **Motivation:** DDPM requires a neural network to predict noise $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)$ while capturing both local details and global image structures.



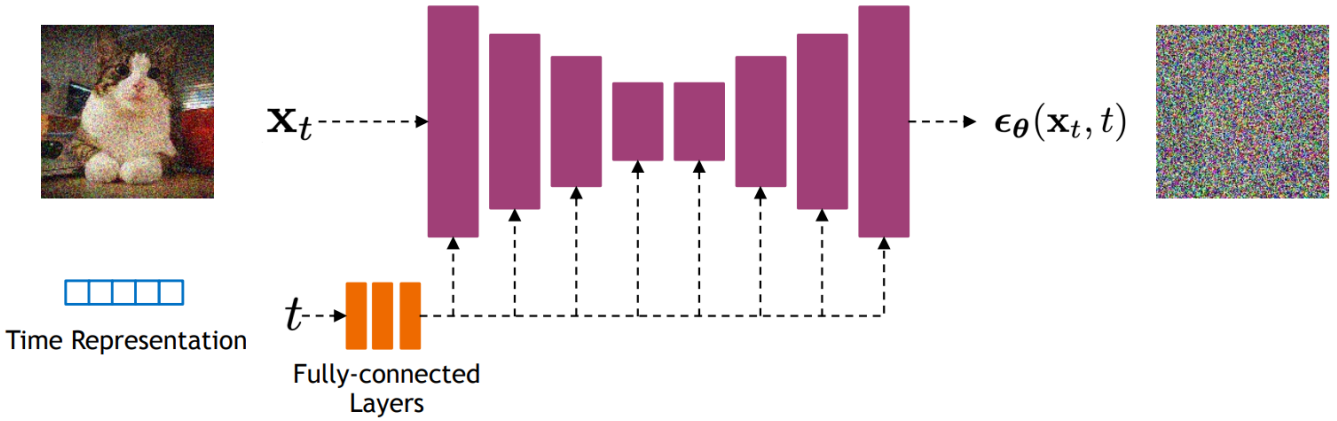
- **Multi-scale Feature Extraction**: The encoder extracts features at different spatial resolutions to capture both fine-grained details and global context.
- **Skip Connections**: Preserve high-resolution spatial information by passing encoder features directly to the decoder.
- **Spatially Aligned Prediction**: The decoder combines multi-scale features to predict noise $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)$ with the same spatial dimensions as the input $\boldsymbol{x}_t$.

Time Encoding

- The time representation is implemented using sinusoidal positional embeddings.
- Given a time step $t \in [T]$ and an embedding dimension d, the sinusoidal positional embedding $\mathrm{SPE}(t)$ is given for $i = 0, 1, \ldots, \frac{d}{2} - 1$ as

$$\mathrm{SPE}(t)_{2i} = \sin\left(\frac{t}{10^{\frac{6i}{d}}}\right), \mathrm{SPE}(t)_{2i+1} = \cos\left(\frac{t}{10^{\frac{6i}{d}}}\right)$$

- This produces a fixed-length vector $\mathrm{SPE}(t)$ that encodes the time smoothly, since small changes in t cause predictable oscillations in the time embedding.
- The time embedding is given as input to all units of the U-Net.



Training and Sampling in DDPM

- DDPM implements the training procedure by performing SGD on the set of training images over timesteps.
- The sampling procedure executes iteratively the denoising process from a Gaussian initialization $\boldsymbol{x}_T$.

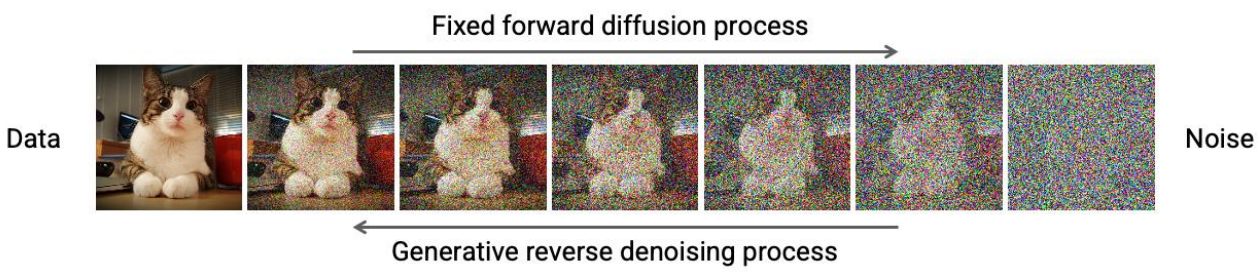
Algorithm 1 Training

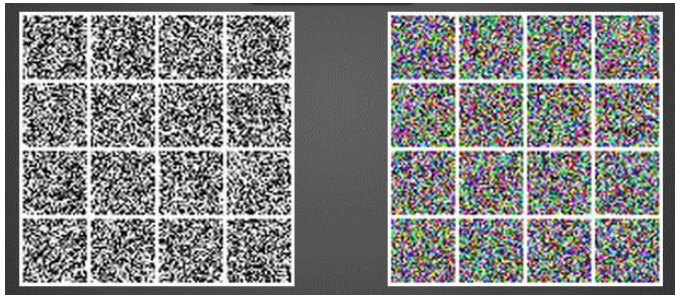
1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0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

$\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t)$ $\quad \sigma_t = \sqrt{\beta_t}$





DDPM Noise Scheduler

- **Noise Scheduler for Forward Process**: in DDPM, the scheduler refers to how the noise variance $(\beta_t = 1 - \alpha_t)$ changes across the diffusion timesteps $t = 1, \dots, T$.
  - **Linear Schedule**: β_t increases linearly from initial value (10^{-4}) to a maximum value (0.02)

$$\beta_t = \beta_{\text{start}} + (t-1)\frac{\beta_{\text{end}} - \beta_{\text{start}}}{T-1}, t = 1, \dots, T$$

  - **Cosine Schedule**: Uses a cosine function to define β_t, which better preserves signal early in the process and decays more gently near the end. It helps maintain more information in the intermediate steps, improve sample quality, require fewer steps for comparable results.

$$f(t) = \cos\left(\frac{t/T+s}{1+s} \cdot \frac{\pi}{2}\right), t = 0, \dots, T, \qquad \overline{\alpha_t} = \frac{f(t)}{f(0)}, \qquad \beta_t = 1 - \frac{\overline{\alpha_t}}{\overline{\alpha_{t-1}}}, \qquad s = 0.008.$$

- **Training Data**: For each training image $\boldsymbol{x}_0$ and timestep t, a noisy image $\boldsymbol{x}_t$ is generated as: $\boldsymbol{x}_t = \sqrt{\overline{\alpha_t}}\boldsymbol{x}_0 + \sqrt{1-\overline{\alpha_t}}\,\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, \boldsymbol{I})$.

Training Curves & Test Metrics

- The Inception Score (IS) and Fréchet Inception Distance (FID) are two commonly used metrics for evaluating the quality of generated images.
- IS measures the quality and diversity of generated samples by comparing the conditional class probability distribution $p(y|\boldsymbol{x})$ with the marginal class probability distribution $p(y)$ using an Inceptionv3 network

$$\mathrm{IS} = \exp\left(\mathbb{E}_{\boldsymbol{x}\sim p_g}\left[D_{\mathrm{KL}}(p(y|\boldsymbol{x}) \,||\, p(y))\right]\right), p(y|\boldsymbol{x}) = \mathrm{InceptionNet}\,(\boldsymbol{x})$$

- The FID measures how close the distribution of the generated images with parameters is $(\boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g)$ to the real data distribution with parameters $(\boldsymbol{\mu}_r, \boldsymbol{\Sigma}_r)$

$$\mathrm{FID} = \left|\left|\boldsymbol{\mu}_r - \boldsymbol{\mu}_g\right|\right|^2 + \mathrm{Tr}\left(\boldsymbol{\Sigma}_r + \boldsymbol{\Sigma}_g - 2\left(\boldsymbol{\Sigma}_r\boldsymbol{\Sigma}_g\right)^{\frac{1}{2}}\right)$$

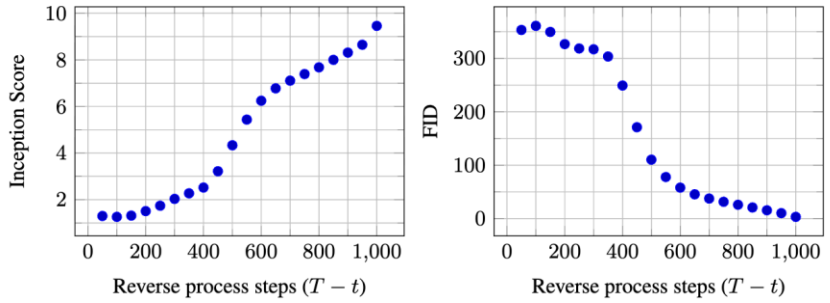


Figure 10: Unconditional CIFAR10 progressive sampling quality over time

Generated Samples of DDPM



