CS 492: Diffusion Models and its applications

Ayush Raina

June 14, 2025

Course Instructor: Prof. Minhyuk Sung
Semester: 7th
Department: Computer Science and Engineering

Contents

1. **Lecture 1:** Course Logistics
2. **Lecture 2:** Introduction to Generative Modelling

Lecture 1

Following is the comparison of DDPM, GAN and VAEs etc:

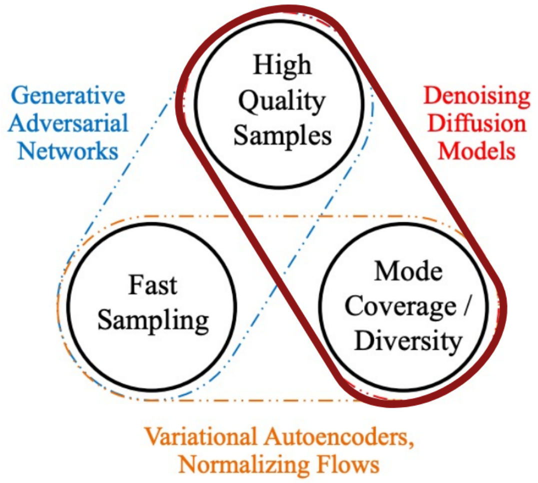

Generative Adversarial Networks
High Quality Samples
Denoising Diffusion Models
Fast Sampling
Mode Coverage / Diversity
Variational Autoencoders, Normalizing Flows

Figure 1: Comparison of DDPM, GAN, and VAEs

Diffusion Models generate high quality samples, cover diversity of data but sampling process is slow as compared to other generative models like GANs.

Following topics will be covered in this course:

1. Introduction to Generative Modelling
2. DDPM and Score Based Models
3. DDIM, Guided Diffusion and Latent Diffusion
4. Control Net, Conditional Generation and Personalization
5. Inverse Problem / Knowledge Distillation
6. Flow Based Models
7. Diffusion Transformers

There will be 7 assignments based on following topics:

1. DDPM
2. DDIM and CFG
3. Control Net and LORA
4. Distillation
5. Synchronization
6. DPM Solvers
7. Flow Based Models

Lecture 2

Sampling from a distribution

To sample from a discrete distribution, we need access to PMF. Suppose we have PMF p, such that $P(X = x_i) = p_i$ for $i = 1, 2, \dots, n$, $p_i \geq 0$ and $\sum_{i=1}^{n} p_i = 1$. Then we can use inverse transform sampling method to sample from this distribution.

Sampling Technique 1: Inverse Transform Sampling

First we calculate the CDF F of the PMF p as follows:

$$F(x_i) = P(X \leq x_i) = \sum_{j=1}^{i} p_j, F(x_n) = 1$$

then we can sample from the distribution as follows:
1. Sample $u \sim \mathcal{U}(0, 1)$.
2. Find i such that $F(x_{i-1}) \leq u \leq F(x_i)$.
3. Return x_i as the sample from the distribution.

In the case of continuous distributions, we can use the same method but with PDF instead of PMF. We can calculate the CDF as follows:

$$F(x) = P(X \leq x) = \int_{-\infty}^{x} p(t)dt$$

Then we can sample from the distribution as follows:
1. Sample $u \sim \mathcal{U}(0, 1)$.
2. Find x such that $F(x) = u$, but this involves the access to the inverse of CDF, which may not be available.
3. Return x as the sample from the distribution.

Example

Consider $p(x) = \frac{3}{8}x^2$ for $0 \leq x \leq 2$. Then we can calculate the CDF as follows:

$$F(x) = \int_{0}^{x} \frac{3}{8}t^2 dt = \frac{1}{8}x^3$$

In this case, we can calculate inverse easily and it turns out to be:

$$F^{-1}(u) = (8u)^{1/3}$$
$$F^{-1}(u) = 2(u)^{1/3}$$

Hence sampling involves 2 steps here:
1. Sample $u \sim \mathcal{U}(0, 1)$.
2. Return $F^{-1}(u) = 2(u)^{1/3}$ as the sample from the distribution.

Sampling Technique 2: Rejection Sampling

If the inverse of the CDF cannot be computed, then:
1. Let $q(x)$ be a proposal (easy-to-sample) distribution and c be a constant such that $c \cdot q(x) \geq p(x)$ for all x.
2. Sample $x \sim q(x)$.
3. Sample $u \sim \mathcal{U}(0, 1)$.
4. Accept the sample x if $u \leq \frac{p(x)}{c \cdot q(x)}$; otherwise reject it and repeat from step 2.

Example: Rejection Sampling

Let's sample from the same distribution $p(x) = \frac{3}{8}x^2$ for $0 \leq x \leq 2$ using rejection sampling.

First, we need to find a proposal distribution $q(x)$ and a constant c such that $c \cdot q(x) \geq p(x)$ for all $x \in [0, 2]$.

Since $p(x) = \frac{3}{8}x^2$ is maximized at $x = 2$, we have $p(2) = \frac{3}{8} \cdot 4 = \frac{3}{2}$. We can choose $q(x) = \mathcal{U}(0, 2)$, which has PDF $q(x) = \frac{1}{2}$ for $x \in [0, 2]$. To ensure $c \cdot q(x) \geq p(x)$, choose $c = 3$, since $c \cdot q(x) = 3 \cdot \frac{1}{2} = \frac{3}{2}$, which matches the maximum of $p(x)$.

The rejection sampling steps are:
1. Sample $x \sim \mathcal{U}(0, 2)$.
2. Sample $u \sim \mathcal{U}(0, 1)$.
3. Accept x if $u \leq \frac{p(x)}{c \cdot q(x)} = \frac{\frac{3}{8}x^2}{3 \cdot \frac{1}{2}} = \frac{x^2}{4}$; otherwise, reject and repeat.

Generative Adversarial Networks (GANs)

Loss Function:

$$\boldsymbol{\min_G \max_D \ \mathcal{L}(D, G) = \mathbb{E}_{x \sim p_{data}}[\log D(x)] + \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))]}$$

Where D is the discriminator, G is the generator, p_{data} is the data distribution, and p_z is the noise distribution. GAN training invoves challenges like **mode collapse** and **non-convergence**.

Non Convergence

Discriminator becomes too good and easily classifies the real and fake samples and as a result leading to zero gradient flow for generator and hence no learning.

Mode Collapse

Generator produces samples from a single mode of the data distribution because for this mode the discriminator is not able to distinguish between real and fake samples. But this does not capture the diversity of the data distribution.

Variational Autoencoders (VAEs)

This is a generative model which does not involve solving the min-max optimization problem. KL Divergence between 2 distributions is defined as:

$$D_{KL}(p\|q) = \int p(x) \log \frac{p(x)}{q(x)} dx = \mathbb{E}_{x \sim p(x)} \left[\log \frac{p(x)}{q(x)} \right]$$

Homework: If $p(x) = \mathcal{N}(x; \mu, \sigma^2 \mathbb{I})$ and $q(x) = \mathcal{N}(x; 0, \mathbb{I})$, then find the KL divergence $D_{KL}(p\|q)$.

Convex Function

A function f is convex if for all x, y and $\lambda \in [0, 1]$:

$$f(\lambda x + (1 - \lambda) y) \leq \lambda f(x) + (1 - \lambda) f(y)$$

In general f is convex if $f(\sum_{i=1}^n \lambda_i x_i) \leq \sum_{i=1}^n \lambda_i f(x_i)$ for all x_i and $\lambda_i \geq 0$ such that $\sum_{i=1}^n \lambda_i = 1$.

Jensen's Inequality

If f is a convex function and X is a random variable, then:

$$\mathbb{E}_{p(x)}[f(X)] \geq f(\mathbb{E}_{p(x)}[X])$$

whereas for concave functions, the inequality is reversed:

$$\mathbb{E}_{p(x)}[f(X)] \leq f(\mathbb{E}_{p(x)}[X])$$

Latent Variable Models introduce hidden (unobserved) variables $\mathbf{z}$ to explain observed data $\mathbf{x}$. The generative process is:

- Sample latent variable: $\mathbf{z} \sim p(\mathbf{z})$
- Generate data: $\mathbf{x} \sim p_\theta(\mathbf{x} \mid \mathbf{z})$

The marginal likelihood of data is:

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x} \mid \mathbf{z}) p(\mathbf{z}) d\mathbf{z}$$

Difference: Previous models directly modeled $p_\theta(\mathbf{x})$; latent variable models introduce $\mathbf{z}$ to capture hidden structure or factors. Learning involves maximizing the marginal likelihood:

$$\theta^* = \arg\max_\theta \sum_{i=1}^N \log \int p_\theta(\mathbf{x}_i \mid \mathbf{z}) p(\mathbf{z}) d\mathbf{z}$$

Problems

We can easily see that this is better modelling choice. We can try to find:

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

$$= \arg\max_\theta \sum_{i=1}^N \log \sum_{z_i \sim \mathcal{Z}} p_\theta(\mathbf{x}_i, z_i)$$

But for a particular x_i : z_i's are not oberserved. Hence we need to marginalize over all possible z_i's which is not feasible. Hence we need some other approach to do MLE estimation on latent variable models.

Evidence Lower Bound (ELBO)

We saw that computing log likelihood of partially observed data $p_\theta(\mathbf{x}, z)$ is hard to compute. We will instead construct a lower bound on LL.

1. Jensen's Inequality: $f(\mathbb{E}[X]) \leq \mathbb{E}[f(X)]$ for convex function f, where as for concave function f we have $f(\mathbb{E}[X]) \geq \mathbb{E}[f(X)]$.

2. log(x) is a concave function. This means:

$$\log(p_\theta(\mathbf{x})) = \log\left(\int p_\theta(\mathbf{x}, z)dz\right) = \log\left(\int q(z)\frac{p_\theta(\mathbf{x}, z)}{q(z)}dz\right)$$

$$= \log\left(\mathbb{E}_{z\sim q(z)}\left[\frac{p_\theta(\mathbf{x}, z)}{q(z)}\right]\right) \geq \mathbb{E}_{z\sim q(z)}\left[\log\left(\frac{p_\theta(\mathbf{x}, z)}{q(z)}\right)\right] := ELBO$$

$\mathcal{L}_{\text{ELBO}}(\theta) := \mathbb{E}_{z\sim q(z)}\left[\log\left(\frac{p_\theta(\mathbf{x},z)}{q(z)}\right)\right]$

We have shown that ELBO is a lower bound on log likelihood and little more math shows that when $q(z) = p_\theta(z|\mathbf{x})$ then equality holds.

Variational Autoencoders (VAEs)

We approximate the posterior $p_\theta(z \mid \mathbf{x})$ with $q_\phi(z \mid \mathbf{x})$ (encoder neural network). Hence our loss function becomes:

$$\mathcal{L}_{\text{VAE}}(\theta, \phi) = \mathbb{E}_{z\sim q_\phi(z|\mathbf{x})}\left[\log\left(\frac{p_\theta(\mathbf{x}, z)}{q_\phi(z \mid \mathbf{x})}\right)\right]$$

$$= \mathbb{E}_{z\sim q_\phi(z|\mathbf{x})}\left[\log(p(z)) + \log(p_\theta(\mathbf{x} \mid z)) - \log(q_\phi(z \mid \mathbf{x}))\right]$$

$$= \mathbb{E}_{z\sim q_\phi(z|\mathbf{x})}\left[\log(p_\theta(\mathbf{x} \mid z)) - \log(\frac{q_\phi(z \mid \mathbf{x})}{p(z)})\right]$$

$$\mathcal{L}_{\text{VAE}}(\theta, \phi) = \mathbb{E}_{z\sim q_\phi(z|\mathbf{x})}\left[\log(p_\theta(\mathbf{x} \mid z))\right] - D_{KL}(q_\phi(z \mid \mathbf{x})||p(z))$$

Here is the final loss function we need to optimize:

$$\mathcal{L}_{\text{VAE}}(\theta, \phi) = \mathbb{E}_{z\sim q_\phi(z|\mathbf{x})}\left[\log(p_\theta(\mathbf{x} \mid z))\right] - D_{KL}(q_\phi(z \mid \mathbf{x})||p(z))$$

1. In VAE literature, $q_\phi(z \mid \mathbf{x})$ is called the **encoder** and $p_\theta(\mathbf{x} \mid z)$ is called the **decoder**.

2. The first term is the **reconstruction loss** (how well the model can reconstruct the data given the latent variable).

3. The second term is the **KL divergence** between the approximate posterior and the prior. It regularizes the latent space to follow a specific distribution (usually Gaussian).

We choose $p(z)$ standard normal distribution $\mathcal{N}(0, I)$ and $q_\phi(z \mid \mathbf{x}) = \mathcal{N}(\mathbf{x}; \mu_\phi(\mathbf{x}), \sigma_\phi(\mathbf{x}))$ and $p_\theta(\mathbf{x} \mid z) = \mathcal{N}(\mathbf{x}; \mu_\theta(z), \sigma_\theta(z))$.

Denoising Diffusion Probabilistic Models (DDPMs)

Today we will finally discuss how all those equations including $\epsilon_\theta(x_t, t)$ and $q(x_{t-1}|x_t)$ etc work together to form a complete generative model.

Consider the forward markovian process with latent dimension same as data dimension with fixed encoding transitions defined as:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t\mathbb{I})$$

where $\{\beta_t\}_{t=1}^T$ are such that $\beta_t \in (0,1)\ \forall t$ and β_t is a monotonically increasing sequence. Above transition kernel is also called **Variance Preserving** form. There are other options like **Variance Exploding** form where the transition kernel is defined as:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; x_{t-1}, (\sigma_i^2 - \sigma_{i-1}^2)\mathbb{I})$$

Choice of β_t

Generally it is linearly or quadratically increased, but can be kept as constant or can be learned as well.

Single and Multi Step Forward Transitions

We defined:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t\mathbb{I})$$

If we choose $\alpha_t = 1 - \beta_t$, then some algebra shows that:

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)\mathbb{I})$$

where $\bar{\alpha}_t = \prod_{i=1}^t \alpha_i$ and it follows that $\{\alpha_t\}_{t=1}^T$ and $\{\bar{\alpha}_t\}_{t=1}^T$ both are monotonically decreasing sequences since $\{\beta_t\}_{t=1}^T$ is monotonically increasing. One can easily verify that $\lim_{T\to\infty} \bar{\alpha}_T = 0$ and due to this reason $q(x_T|x_0)$ converges to a Gaussian with mean 0 and variance $\mathbb{I}$.

ELBO Objective Again

In this case, the negative of ELBO can be expressed as:

$$-\log p_\theta(x_0) = -\log \int p_\theta(x_{0:T})dx_{1:T} \geq \mathbb{E}_{q(x_{1:T}|x_0)}\left[-\log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}\right] := \mathcal{L}(\theta)$$

This is very similar to ELBO in VAE case. After some algebra this can be broken into 3 terms:

- **Reconstruction Loss:**
$$\mathcal{L}_{rec}(\theta) = -\mathbb{E}_{q(x_1|x_0)}\left[\log p_\theta(x_0|x_1)\right]$$
- **Prior Matching Loss:**
$$\mathcal{L}_{prior}(\theta) = \mathbb{E}_{q(x_{T-1}|x_0)}\ \left[\mathcal{D}_{KL}(q(x_T|x_{T-1})\ ||\ p(x_T))\right]$$
- **Consistency term:**
$$\mathcal{L}_{consistency}(\theta) = \sum_{t=1}^{T-1} \mathbb{E}_{q(x_{t-1,t+1}|x_0)}\left[\mathcal{D}_{KL}(q(x_t|x_{t-1})||p_\theta(x_t|x_{t+1}))\right]$$

Remember we have not yet derived any equations, so keep going in this flow only to understand everthing. From consistency term we can see that expectation is over 2 random variables which is difficult to compute. But if we decompose ELBO using $q(x_t|x_{t-1}, x_0)$ instead of $q(x_t|x_{t-1})$ we get simpler form even though both are equivalent. On decomposing ELBO again, reconstruction term remains same, Prior Matching term becomes 0 because we choose noise schedule in a way that $q(x_T) = p(x_T)$, and consistency term becomes:

$$\mathcal{L}_{consistency}(\theta) = \sum_{t=2}^{T} \mathbb{E}_{q(x_t|x_0)}\left[\mathcal{D}_{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t))\right]$$

Let us call this as **Denoising Matching Term** and our goal is now to minimize this term.

Now we can observe that we need to calculate $\mathcal{D}_{KL}(q(x_{t-1}|x_t,x_0)||p_\theta(x_{t-1}|x_t))$ and for this we need to know the form of both distributions involved there. Applying Bayes rule to $q(x_{t-1}|x_t,x_0)$ we get:

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

We know all the three terms involved here and they are as follows:

- $q(x_t|x_{t-1},x_0) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t\mathbb{I})$
- $q(x_{t-1}|x_0) = \mathcal{N}(x_{t-1}; \sqrt{\bar{\alpha}_{t-1}}x_0, (1-\bar{\alpha}_{t-1})\mathbb{I})$
- $q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)\mathbb{I})$

After substituting these 3 terms and doing some long algebra we get:

$$q(x_{t-1}|x_t,x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}_t(x_t,x_0), \tilde{\sigma}_t^2\mathbb{I})$$

where

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

and

$$\tilde{\sigma}_t^2 = \frac{(1-\bar{\alpha}_{t-1})\beta_t}{1-\bar{\alpha}_t}$$

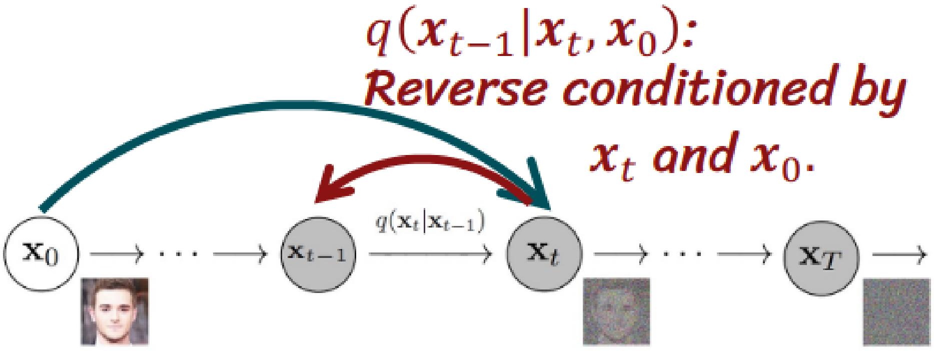


Figure 2: Denoising Matching Term

We can see if x_0, x_t are available to us then we can use this formulation. There are other formulations as well which we will now discuss.

Formulation 1: Mean Predictor Network $\mu_\theta(x_t,t)$

In this formulation, we model $p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t,t), \tilde{\sigma}^2\mathbb{I})$, variance schedule is same in both $q(x_{t-1}|x_t,x_0)$ and $p_\theta(x_{t-1}|x_t)$ since we know that. In this case, we can write the Denoising Matching Term as:

$$\mathbb{E}_{x_0\sim q(x_0),t>1,x_t\sim q(x_t|x_0)}\left[\frac{1}{2\tilde{\sigma}_t^2}\left\|\tilde{\mu}_\theta(x_t,x_0) - \mu_\theta(x_t,t)\right\|^2\right]$$

Formulation 2: x_0 **Predictor Network** $\hat{x}_\theta(x_t,t)$

In this formulation, we model:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}x_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\hat{x}_\theta(x_t,t), \tilde{\sigma}_t^2\mathbb{I})$$

and in this case we can write the Denoising Matching Term as:

$$\mathbb{E}_{x_0\sim q(x_0),t>1,x_t\sim q(x_t|x_0)}\left[\frac{1}{2\tilde{\sigma}_t^2}\omega_t\left\|\hat{x}_\theta(x_t,t) - x_0\right\|^2\right]$$

Formulation 3: $\epsilon_\theta(x_t, t)$ **Predictor Network**

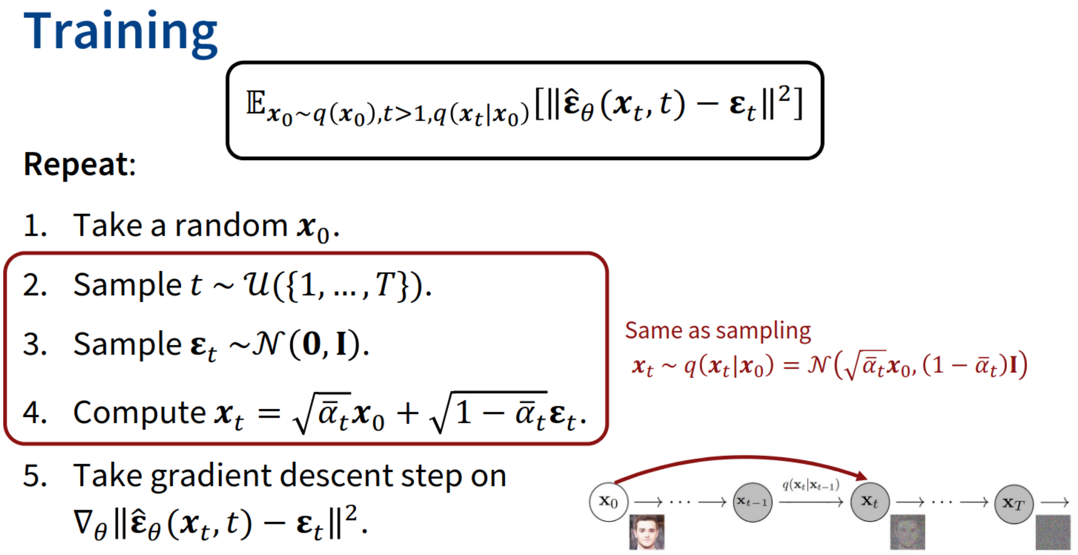
In this formulation, we model:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)), \tilde{\sigma}_t^2\mathbb{I})$$

and in this case we can write the Denoising Matching Term as:

$$\mathbb{E}_{x_0\sim q(x_0),t>1,x_t\sim q(x_t|x_0)}\left[\frac{1}{2\tilde{\sigma}_t^2}\omega_t'\left\|\epsilon_\theta(x_t,t)-\epsilon_t\right\|^2\right]$$

In practice we drop the the scaling terms and generally 3rd formulation works better. Here are the training and inference algorithms for DDPMs:



Training

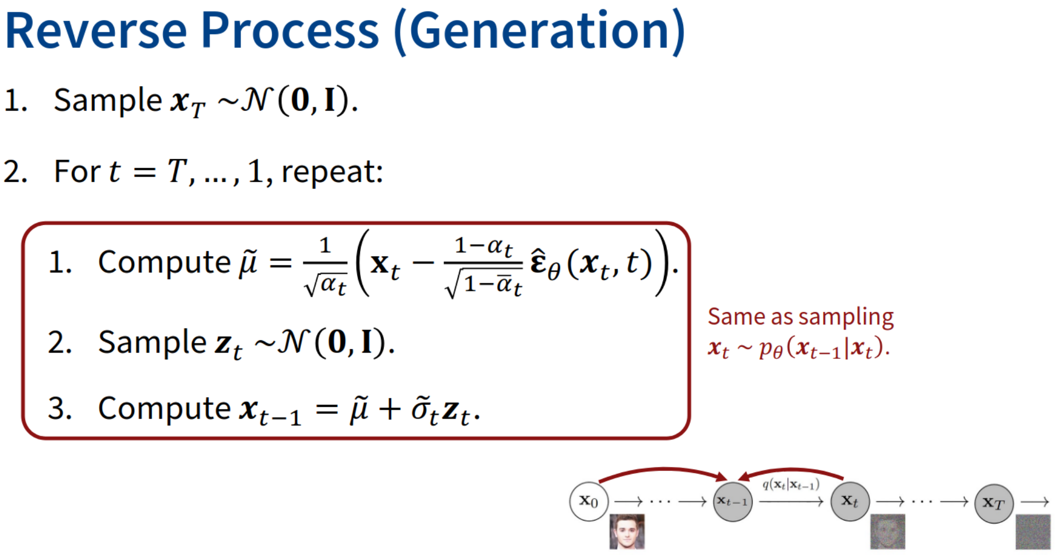
$\mathbb{E}_{x_0\sim q(x_0),t>1,q(x_t|x_0)}[||\hat{\varepsilon}_\theta(x_t,t)-\varepsilon_t||^2]$

Repeat:

1. Take a random x_0.
2. Sample $t \sim \mathcal{U}(\{1,\ldots,T\})$.
3. Sample $\varepsilon_t \sim \mathcal{N}(\mathbf{0},\mathbf{I})$.
4. Compute $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon_t$.
5. Take gradient descent step on $\nabla_\theta||\hat{\varepsilon}_\theta(x_t,t)-\varepsilon_t||^2$.

Same as sampling $x_t \sim q(x_t|x_0) = \mathcal{N}(\sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)\mathbf{I})$

Figure 3: Training Algorithm for DDPMs



Reverse Process (Generation)

1. Sample $x_T \sim \mathcal{N}(\mathbf{0},\mathbf{I})$.
2. For $t = T,\ldots,1$, repeat:
   1. Compute $\tilde{\mu} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\hat{\varepsilon}_\theta(x_t,t)\right)$.
   2. Sample $\mathbf{z}_t \sim \mathcal{N}(\mathbf{0},\mathbf{I})$.
   3. Compute $x_{t-1} = \tilde{\mu} + \tilde{\sigma}_t\mathbf{z}_t$.

Same as sampling $x_t \sim p_\theta(x_{t-1}|x_t)$.

Figure 4: Inference Algorithm for DDPMs