

Diffusion Models

Machine Learning Algorithms

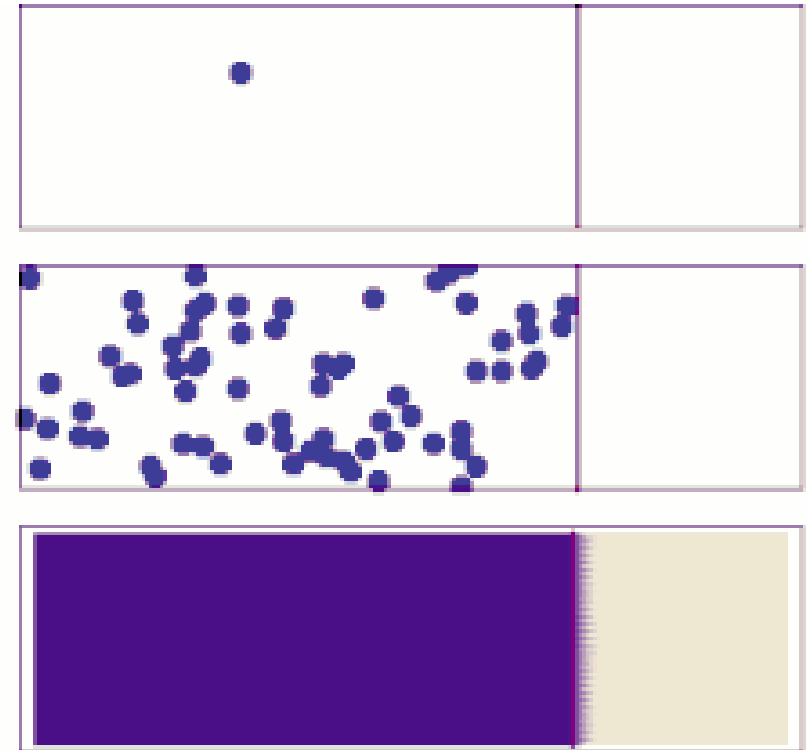
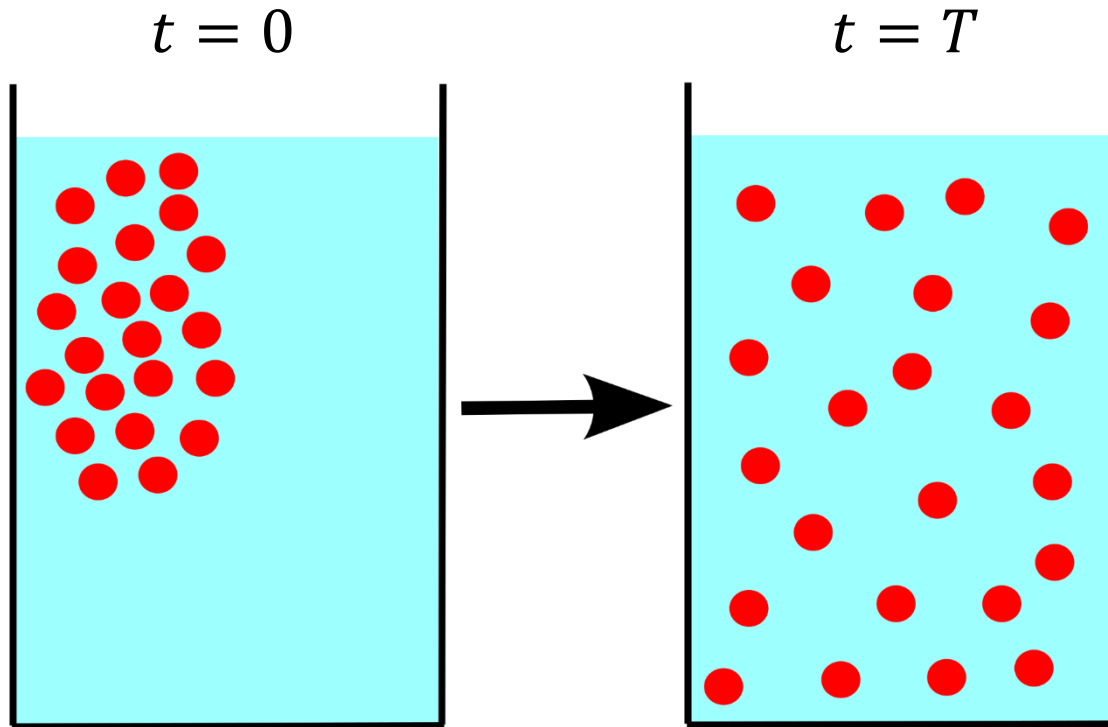
Cheongjae Jang

cjjang@hanyang.ac.kr

Many of the contents and figures are obtained from Bishop (2023).

Diffusion

“**Diffusion** is the net movement of anything (for example, atoms, ions, molecules, energy) generally from a region of higher concentration to a region of lower concentration.”



Diffusion for Probabilistic Modeling of Data

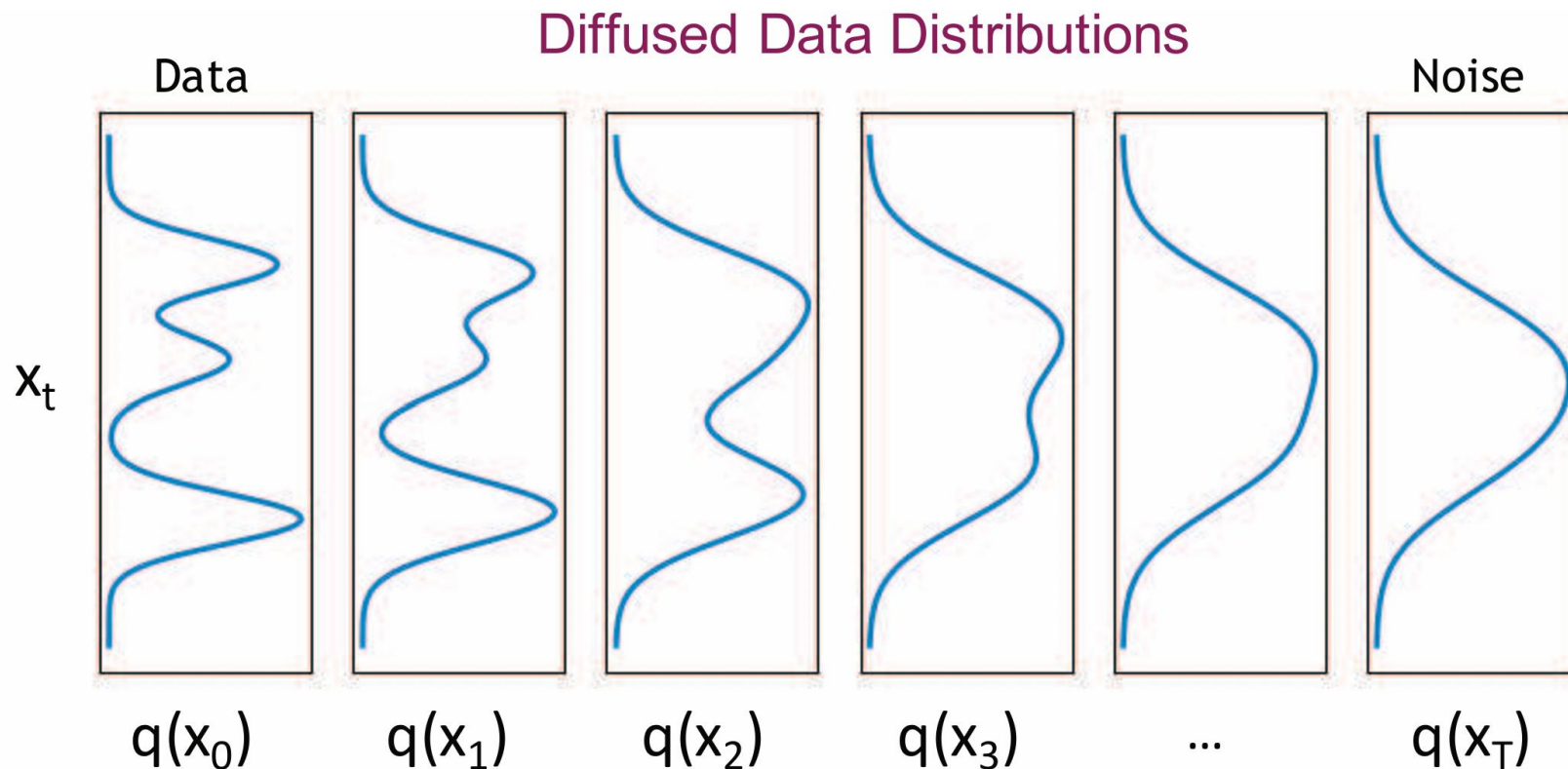


Figure 25.2: Illustration of a diffusion model on 1d data. The forwards diffusion process gradually transforms the empirical data distribution $q(\mathbf{x}_0)$ into a simple target distribution, here $q(\mathbf{x}_T) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. To generate from the model, we sample a point $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and then run the Markov chain backwards, by sampling $\mathbf{x}_t \sim p_{\theta}(\mathbf{x}_t | \mathbf{x}_{t+1})$ until we get a sample in the original data space, \mathbf{x}_0 . From Slide 19 of [KGV22]. Used with kind permission of Arash Vahdat.

Denoising Diffusion Probabilistic Models

Jonathan Ho

UC Berkeley

jonathanho@berkeley.edu

Ajay Jain

UC Berkeley

ajayj@berkeley.edu

Pieter Abbeel

UC Berkeley

pabbeel@cs.berkeley.edu

Abstract

We present high quality image synthesis results using diffusion probabilistic models, a class of latent variable models inspired by considerations from nonequilibrium thermodynamics. Our best results are obtained by training on a weighted variational bound designed according to a novel connection between diffusion probabilistic models and denoising score matching with Langevin dynamics, and our models naturally admit a progressive lossy decompression scheme that can be interpreted as a generalization of autoregressive decoding. On the unconditional CIFAR10 dataset, we obtain an Inception score of 9.46 and a state-of-the-art FID score of 3.17. On 256x256 LSUN, we obtain sample quality similar to ProgressiveGAN. Our implementation is available at <https://github.com/hojonathanho/diffusion>.

NeurIPS 2020

DDPM (2020)

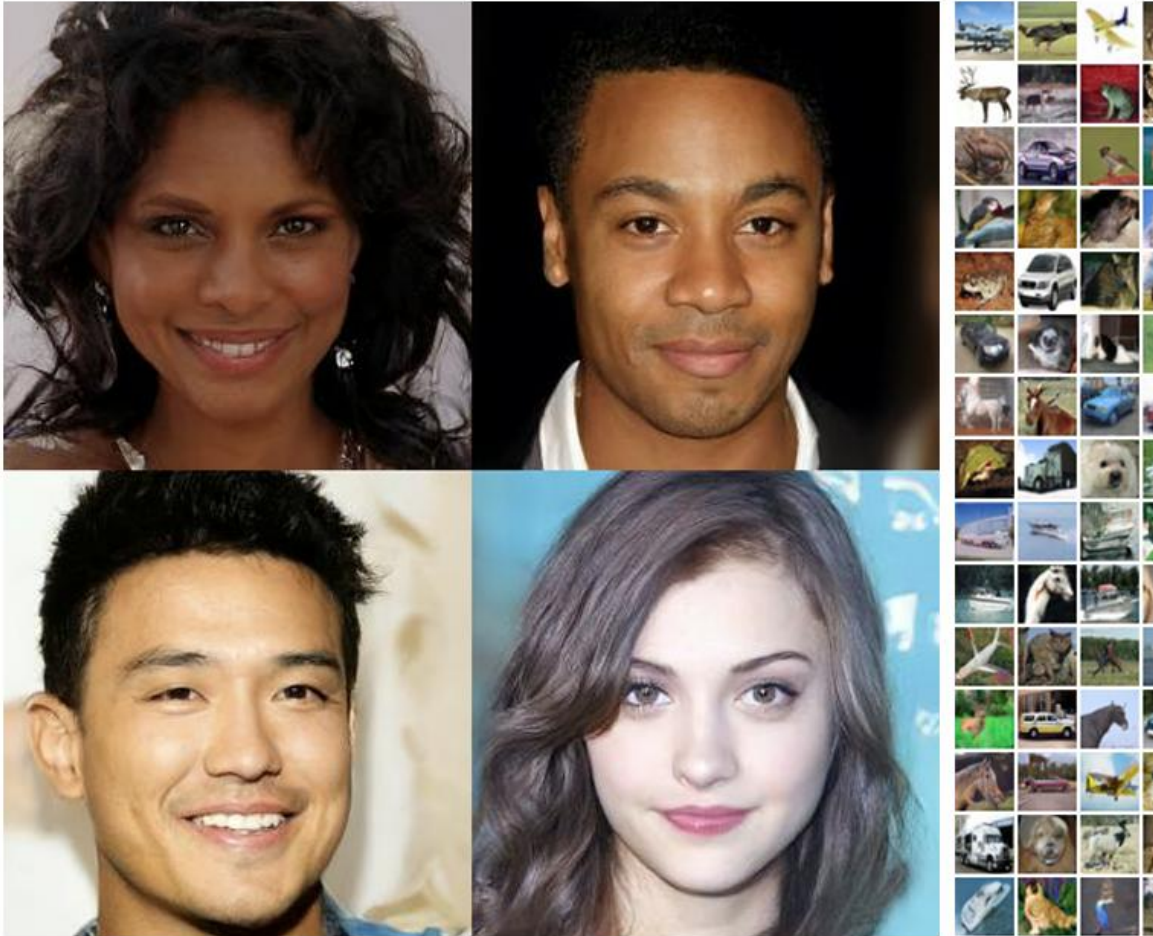
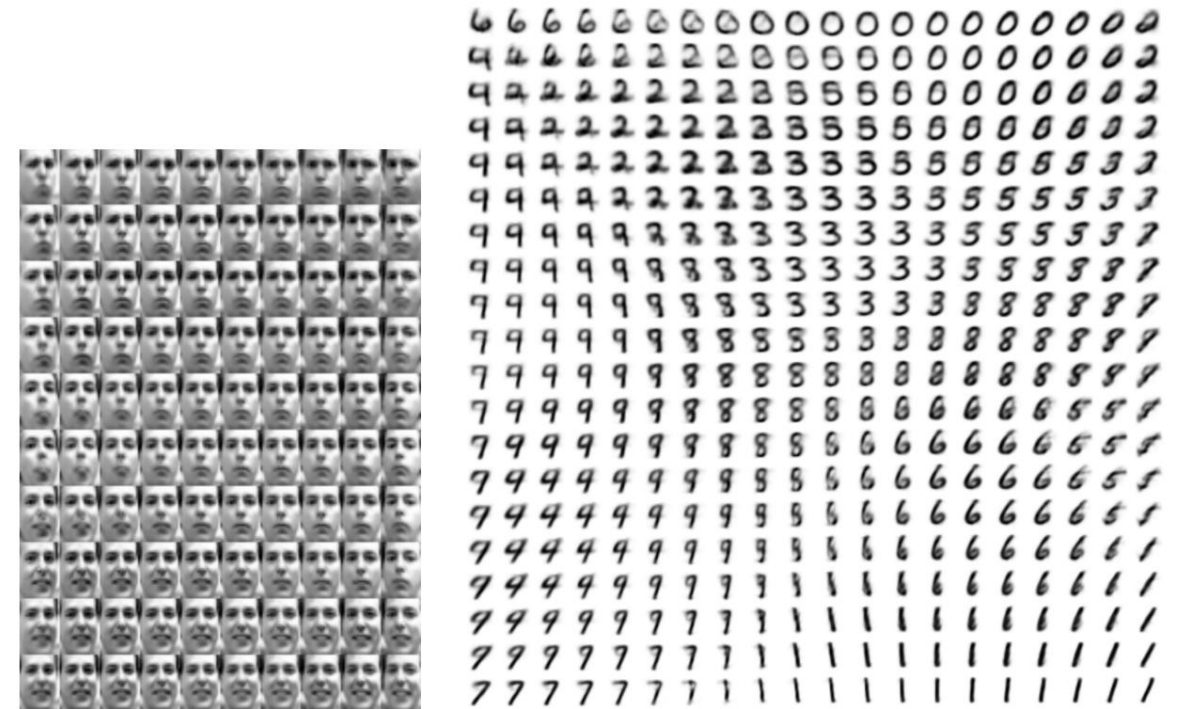


Figure 1: Generated samples on CelebA-HQ 256×256 (left) and uncond

VAE (2014)



(a) Learned Frey Face manifold

(b) Learned MNIST manifold

Figure 4: Visualisations of learned data manifold for generative models with two-dimensional latent space, learned with AEVB. Since the prior of the latent space is Gaussian, linearly spaced coordinates on the unit square were transformed through the inverse CDF of the Gaussian to produce values of the latent variables \mathbf{z} . For each of these values \mathbf{z} , we plotted the corresponding generative $p_{\theta}(\mathbf{x}|\mathbf{z})$ with the learned parameters θ .

The Basic Idea

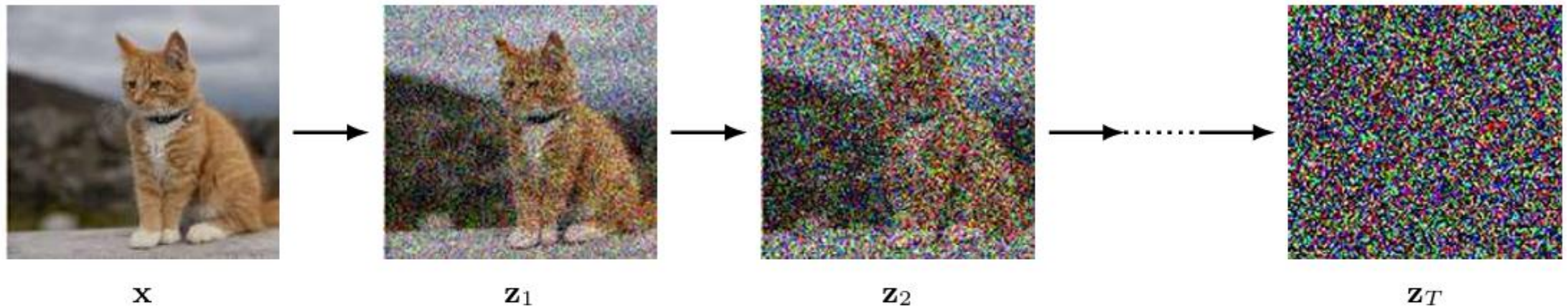
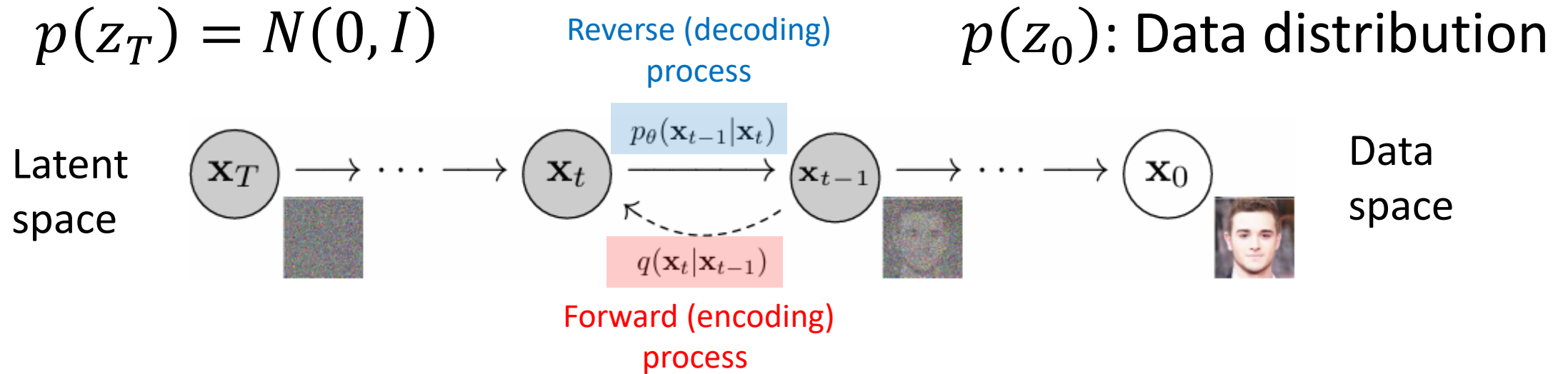


Figure 20.1 Illustration of the encoding process in a diffusion model showing an image x that is gradually corrupted with multiple stages of additive Gaussian noise giving a sequence of increasingly noisy images. After a large number T of steps the result is indistinguishable from a sample drawn from a Gaussian distribution. A deep neural network is then trained to reverse this process.

The Basic Idea



Diffusion models can be viewed as a form of **hierarchical VAE**, where the **encoder distribution** is fixed and defined by the noise process, and only the **generative distribution** is learned.

Forward Encoder

x : an image from the training set

A noise-corrupted image from x :

$$z_1 = \sqrt{1 - \beta_1}x + \sqrt{\beta_1}\epsilon_1$$

$$q(z_1|x) = N(z_1|\sqrt{1 - \beta_1}x, \beta_1 I)$$

$$\epsilon_1 \sim N(\epsilon_1|0, I)$$

$\beta_1 < 1$: the variance of the noise distribution

At time step t :

$$z_t = \sqrt{1 - \beta_t}z_{t-1} + \sqrt{\beta_t}\epsilon_t$$

$$q(z_t|z_{t-1}) = N(z_t|\sqrt{1 - \beta_t}z_{t-1}, \beta_t I)$$

$$\epsilon_t \sim N(\epsilon_t|0, I)$$

$$\beta_t \in (0, 1)$$

Usual choices: $\beta_1 < \beta_2 < \dots < \beta_T$

Forward process (similar to the encoder of VAE)

Forward Encoder

$$z_t = \sqrt{1 - \beta_t} z_{t-1} + \sqrt{\beta_t} \epsilon_t$$

$$\epsilon_t \sim N(\epsilon_t | 0, I)$$

$$\beta_t \in (0, 1)$$

Q. What if we use
 $z_t = z_{t-1} + \sqrt{\beta_t} \epsilon_t$?

$$\mathbb{E}[z_t] = \sqrt{1 - \beta_t} \mathbb{E}[z_{t-1}]$$

→ The **mean** of z_t will be closer to zero than that of z_{t-1} .

$$\text{Cov}[z_t] = (1 - \beta_t) \cdot \text{Cov}[z_{t-1}] + \beta_t \cdot I$$

→ The **covariance** of z_t will be closer to the identity matrix than that of z_{t-1} .

Forward Encoder

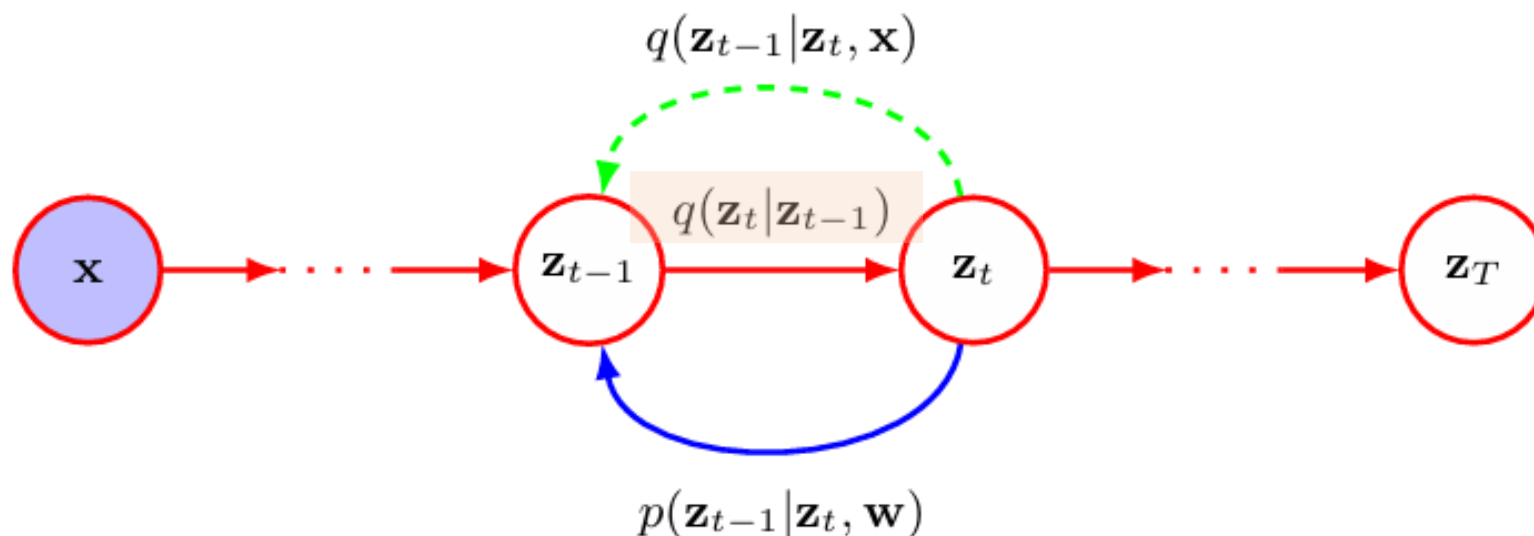


Figure 20.2 A diffusion process represented as a probabilistic graphical model. The original image \mathbf{x} is shown by the shaded node, since it is an observed variable, whereas the noise-corrupted images $\mathbf{z}_1, \dots, \mathbf{z}_T$ are considered to be latent variables. The noise process is defined by the forward distribution $q(\mathbf{z}_t | \mathbf{z}_{t-1})$ and can be viewed as an encoder. Our goal is to learn a model $p(\mathbf{z}_{t-1} | \mathbf{z}_t, \mathbf{w})$ that tries to reverse this noise process and which can be viewed as a decoder. As we will see later, the conditional distribution $q(\mathbf{z}_{t-1} | \mathbf{z}_t, \mathbf{x})$ plays an important role in defining the training procedure.

Diffusion Kernel

From the factorization of the joint PDF of a directed graphical model:

$$q(z_1, \dots, z_t | x) = q(z_1 | x) \prod_{\tau=2}^t q(z_\tau | z_{\tau-1}) \quad \longrightarrow \quad \text{This **conditional** distribution is jointly Gaussian!}$$

Marginalize over z_1, \dots, z_{t-1} to obtain the **diffusion kernel**:

$$q(z_t | x) = N(z_t | \sqrt{\alpha_t}x, (1 - \alpha_t)I)$$

where $\alpha_t = \prod_{\tau=1}^t (1 - \beta_\tau).$

Is this expression correct?

Prove it as an exercise ☺



$$z_t = \sqrt{\alpha_t}x + \sqrt{1 - \alpha_t}\epsilon_t$$

$$\epsilon_t \sim N(\epsilon_t | 0, I)$$

Note: This expression models z_t from the **original image** x and a **global noise** ϵ_t . This will be useful for **training DDPMs** later...

Diffusion Kernel

After many steps, the image becomes indistinguishable from the Gaussian noise.

In the limit $T \rightarrow \infty$,
$$\alpha_T = \prod_{\tau=1}^T (1 - \beta_\tau) \rightarrow 0.$$

$$q(z_T|x) = N(z_t | \sqrt{\alpha_T}x, (1 - \alpha_T)I) \quad \longrightarrow \quad q(z_T|x) = N(z_T | 0, I)$$

All information about the original image is **lost**.

$q(z_T|x)$ is independent of $x \rightarrow$ The marginal PDF $q(z_T) = N(z_T | 0, I)$.

Conditional Distribution

Q. How can we **reverse** the noise process?

Let's consider the **Bayes rule** for $q(z_{t-1}|z_t)$.

$$q(z_{t-1}|z_t) = \frac{q(z_t|z_{t-1})q(z_{t-1})}{q(z_t)}$$

where $q(z_{t-1}) = \int q(z_{t-1}|x)p(x) dx$ and $q(z_t|x) = N(z_t|\sqrt{\alpha_t}x, (1 - \alpha_t)I)$.

This distribution is **intractable!**

- Unknown data density $p(x)$.
- The Monte Carlo approximation will give a complicated distribution as a **mixture of Gaussians**.

Conditional Distribution

Q. How can we **reverse** the noise process?

Let's consider the **Bayes rule** for the **conditional version** $q(z_{t-1}|z_t, x)$.

(Intuitively, denoising will be much easier if we know the original image x .)

$$q(z_{t-1}|z_t, x) = \frac{q(z_t|z_{t-1}, x)q(z_{t-1}|x)}{q(z_t|x)}$$

These are
exponentials of a
quadratic form of
 z_{t-1} .

where $q(z_t|z_{t-1}, x) = q(z_t|z_{t-1})$ (according to the Markov property of the graphical model)

$$= N(z_t|\sqrt{1-\beta_t}z_{t-1}, \beta_t I)$$

and $q(z_{t-1}|x) = N(z_{t-1}|\sqrt{\alpha_{t-1}}x, (1-\alpha_{t-1})I).$

➡ $q(z_{t-1}|z_t, x)$ is a **Gaussian!** $q(z_{t-1}|z_t, x) = N(z_{t-1}|m_t(x, z_t), \sigma_t^2 I)$

Conditional Distribution

$$q(z_{t-1}|z_t, x) = N(z_{t-1}|m_t(x, z_t), \sigma_t^2 I)$$

where

$$m_t(x, z_t) = \frac{(1 - \alpha_{t-1})\sqrt{1 - \beta_t}z_t + \sqrt{\alpha_{t-1}}\beta_t x}{1 - \alpha_t}$$

and

$$\sigma_t^2 = \frac{\beta_t(1 - \alpha_{t-1})}{1 - \alpha_t}.$$

Is this expression correct? Prove it as an exercise 😊

Reverse Decoder

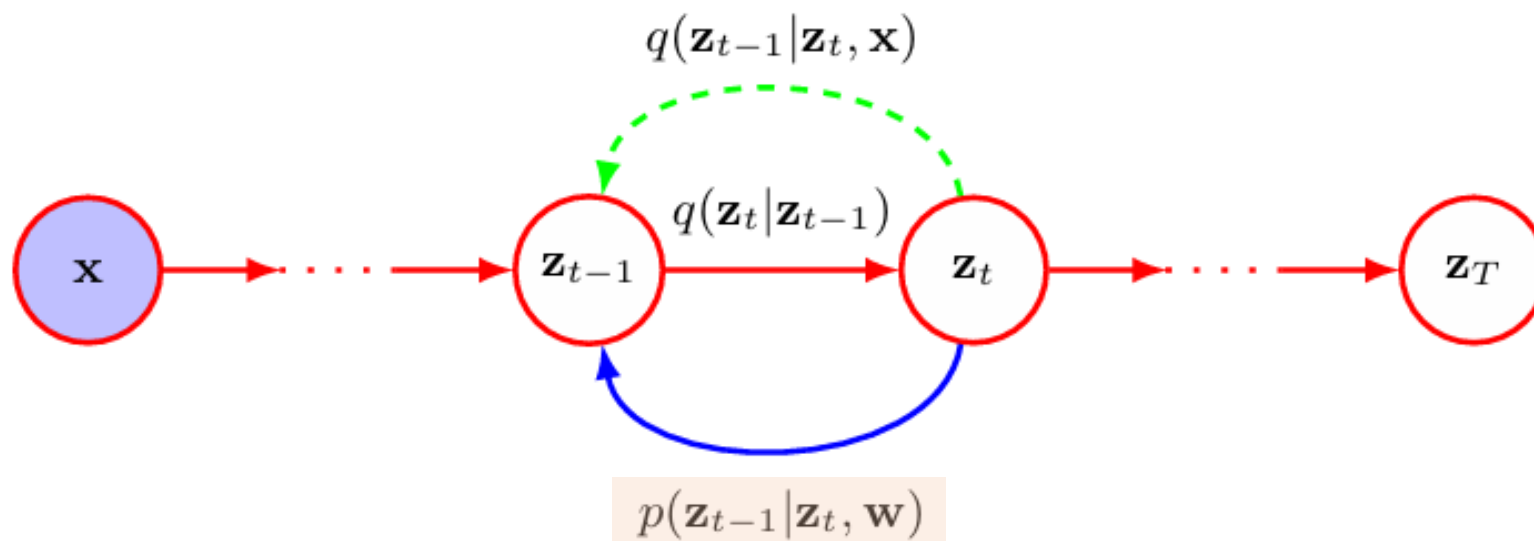


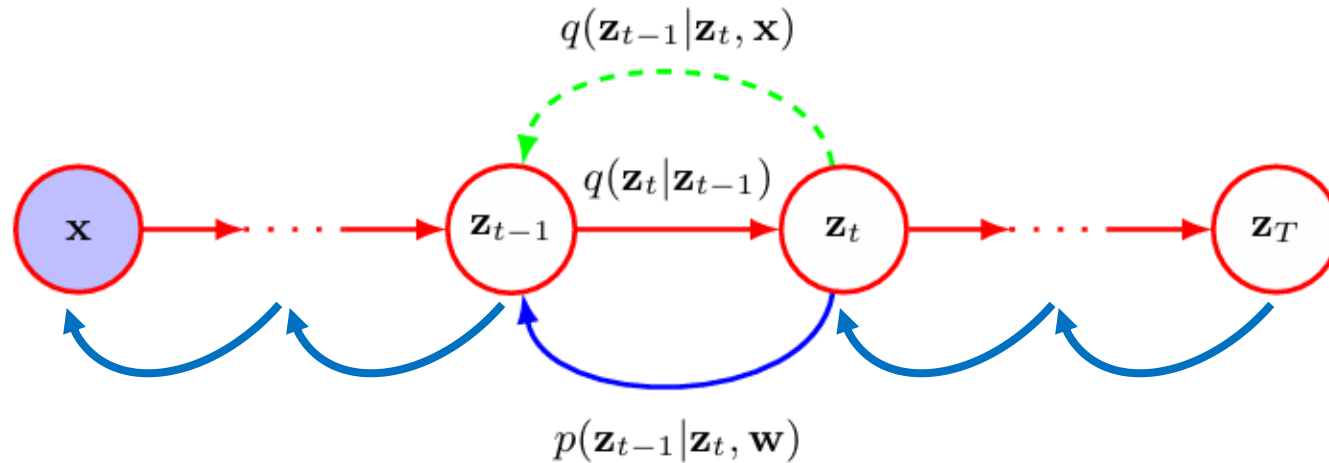
Figure 20.2 A diffusion process represented as a probabilistic graphical model. The original image \mathbf{x} is shown by the shaded node, since it is an observed variable, whereas the noise-corrupted images $\mathbf{z}_1, \dots, \mathbf{z}_T$ are considered to be latent variables. The noise process is defined by the forward distribution $q(\mathbf{z}_t | \mathbf{z}_{t-1})$ and can be viewed as an encoder. Our goal is to learn a model $p(\mathbf{z}_{t-1} | \mathbf{z}_t, \mathbf{w})$ that tries to reverse this noise process and which can be viewed as a decoder. As we will see later, the conditional distribution $q(\mathbf{z}_{t-1} | \mathbf{z}_t, \mathbf{x})$ plays an important role in defining the training procedure.

Reverse Decoder

Recall that $q(z_{t-1}|z_t)$ is intractable (due to the integration involving $p(x)$).

Let's learn **an approximation** $p(z_{t-1}|z_t, w)$ for the reverse distribution $q(z_{t-1}|z_t)$!

$p(z_{t-1}|z_t, w)$ is modeled by a **deep neural network**, which can be considered as a flexible function with parameters w .



$$p(z_T) = N(z_T|0, I)$$

We can **sample data** through a **sequence of reverse sampling steps** by repeated application of the trained network.

Reverse Decoder

Model the reverse process using a **Gaussian distribution**:

$$p(z_{t-1}|z_t, w) = N(z_{t-1}|\mu(z_t, w, t), \beta_t I)$$

Q. When will $q(z_{t-1}|z_t)$ be **similar to a Gaussian**?

$$q(z_{t-1}|z_t) = \frac{q(z_t|z_{t-1})q(z_{t-1})}{q(z_t)} \quad \text{where} \quad q(z_t|z_{t-1}) = N(z_t|\sqrt{1-\beta_t}z_{t-1}, \beta_t I).$$

If $\beta_t \ll 1$, then $q(z_t|z_{t-1})$ will be a **narrow** Gaussian.

If we see the **numerator as a function of** z_{t-1} , then the values of z_{t-1} that yield **high values** of $q(z_t|z_{t-1})$ will be concentrated in a small region.

The value of $q(z_{t-1})$ will **vary only a small amount** in that region.

→ $q(z_{t-1}|z_t)$ can be approximately Gaussian when $\beta_t \ll 1$.

(In this case, we will require a large number of steps T , typically several thousand.)

Reverse Decoder

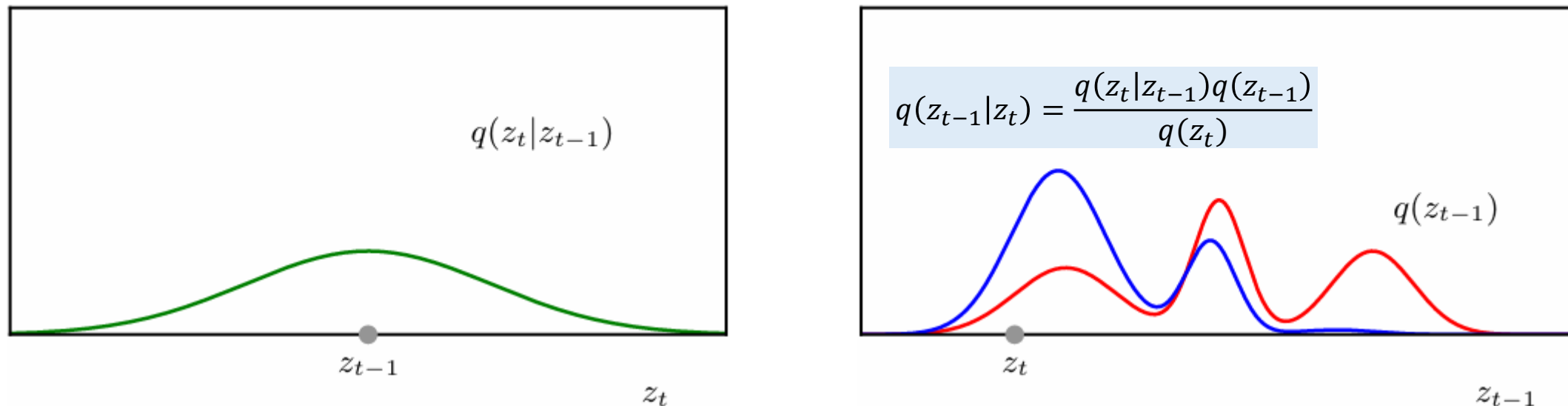


Figure 20.3 Illustration of the evaluation of the reverse distribution $q(z_{t-1}|z_t)$ using Bayes' theorem (20.13) for scalar variables. The red curve on the right-hand plot shows the marginal distribution $q(z_{t-1})$ illustrated using a mixture of three Gaussians, whereas the left-hand plot shows the Gaussian forward noise process $q(z_t|z_{t-1})$ as a distribution over z_t centred on z_{t-1} . By multiplying these together and normalizing, we obtain the distribution $q(z_{t-1}|z_t)$ shown for a particular choice of z_t by the blue curve. Because the distribution on the left is relatively broad, corresponding to a large variance β_t , the distribution $q(z_{t-1}|z_t)$ has a complex multimodal structure.

Reverse Decoder

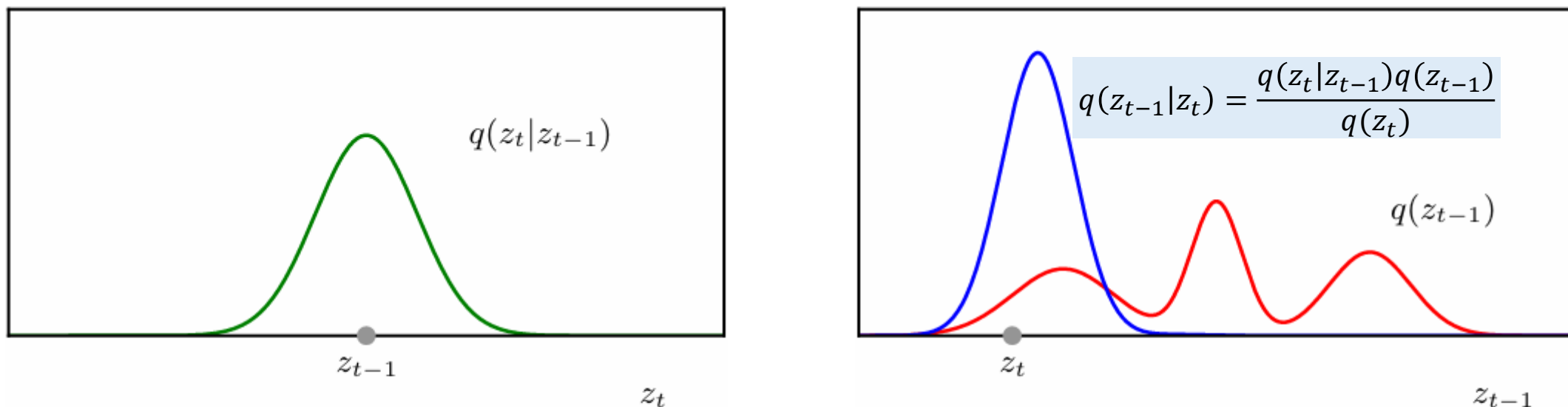


Figure 20.4 As in Figure 20.3 but in which the Gaussian distribution $q(z_t | z_{t-1})$ in the left-hand plot has a much smaller variance β_t . We see that the corresponding distribution $q(z_{t-1} | z_t)$ shown in blue on the right-hand plot is close to being Gaussian, with a similar variance to $q(z_t | z_{t-1})$.

Reverse Decoder

Model the reverse process using a **Gaussian distribution**:

$$p(z_{t-1}|z_t, w) = N(z_{t-1}|\mu(z_t, w, t), \beta_t I)$$

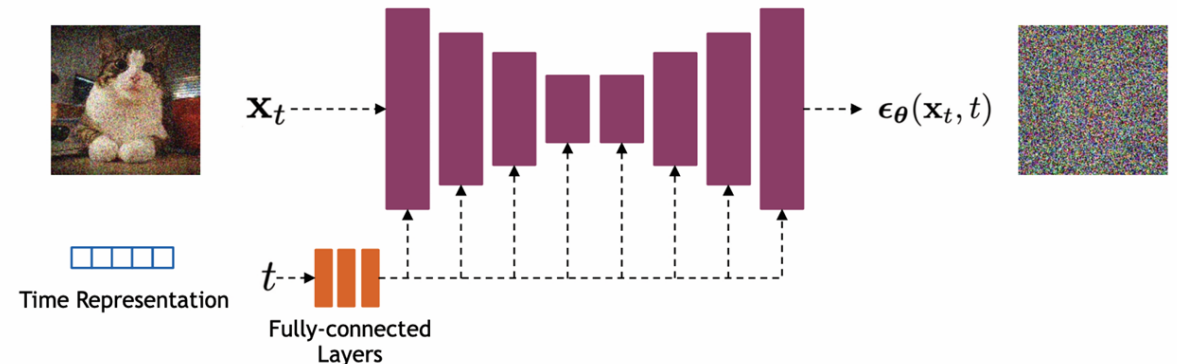
$\beta_t I$: the covariance of the forward noise process

$\mu(z_t, w, t)$: a deep neural network with a set of parameters w

t : step index

Using the **step index** t as an **input** to $\mu(z_t, w, t)$ allows us to **use a single network to invert all the steps**, instead of having to learn a separate network for each step.

$\mu(z_t, w, t)$ should provide the output which has the **same dimensionality** as the input.
→ A neural network architecture called **U-net** is a common choice for **image processing applications**.



Reverse Decoder

The overall **denoising process**:

$$p(x, z_1, \dots, z_T | w) = p(z_T) \left(\prod_{t=2}^T p(z_{t-1} | z_t, w) \right) p(x | z_1, w)$$

$$p(z_T) = q(z_T) = N(z_T | 0, I)$$

We first sample from the Gaussian $p(z_T)$.

Sample sequentially from each of the conditional distributions $p(z_{t-1} | z_t, w)$ in turn.

Finally, sample from $p(x | z_1, w)$ to obtain a sample x in the data space.

Training the Decoder

Consider the likelihood function:

$$p(x|w) = \int \cdots \int p(x, z_1, \dots, z_T|w) dz_1 \dots dz_T$$

This integral is **intractable**, because we have to integrate over the highly complex neural network functions.

Similarly to the case of variational autoencoders (VAE), we **maximize a lower bound on the log likelihood** called the **evidence lower bound (ELBO)**.

Evidence Lower Bound

$$\begin{aligned}\log p(x|w) &= \int q(z) \log p(x|w) dz \\ &= \int q(z) \log \left(\frac{p(x, z|w)}{p(z|x, w)} \cdot \frac{q(z)}{q(z)} \right) dz \\ &= \int q(z) \log \left(\frac{p(x, z|w)}{q(z)} \right) + q(z) \log \left(\frac{q(z)}{p(z|x, w)} \right) dz \\ &= L(w) + KL(q(z) || p(z|x, w))\end{aligned}$$

$$\text{where } L(w) = \int q(z) \log \left(\frac{p(x, z|w)}{q(z)} \right) dz \quad \text{and} \quad KL(f(z) || g(z)) = \int f(z) \log \left(\frac{f(z)}{g(z)} \right) dz.$$

As $KL(\cdot || \cdot) \geq 0$, we have $\log p(x|w) \geq L(w)$.

Since the **log likelihood function is intractable**, we train the model by **maximizing the lower bound**.

Evidence Lower Bound


Using any PDF of z for $q(z)$ satisfies the inequality, and in diffusion models, we choose $q(z)$ to be the **fixed** distribution $q(z_1, \dots, z_T | x)$.

$$\begin{aligned}
 L(w) &= \int q(z) \log \left(\frac{p(x, z | w)}{q(z)} \right) dz \\
 &= \mathbb{E}_q \left[\log \frac{p(z_T) \left(\prod_{t=2}^T p(z_{t-1} | z_t, w) \right) p(x | z_1, w)}{q(z_1 | x) \prod_{t=2}^T q(z_t | z_{t-1}, x)} \right] \\
 &= \mathbb{E}_q \left[\log p(z_T) + \sum_{t=2}^T \log \frac{p(z_{t-1} | z_t, w)}{q(z_t | z_{t-1}, x)} - \log q(z_1 | x) + \log p(x | z_1, w) \right]
 \end{aligned}$$

$p(x, z_1, \dots, z_T | w) = p(z_T) \left(\prod_{t=2}^T p(z_{t-1} | z_t, w) \right) p(x | z_1, w)$
 $q(z_1, \dots, z_T | x) = q(z_1 | x) \prod_{t=2}^T q(z_t | z_{t-1}, x)$

$\mathbb{E}_q[\cdot] = \int \dots \int q(z_1 | x) \prod_{t=2}^T q(z_t | z_{t-1}) [\cdot] dz_1 \dots dz_T$

Evidence Lower Bound

$$L(w) = \mathbb{E}_q \left[\underbrace{\log p(z_T)} + \sum_{t=2}^T \log \frac{p(z_{t-1}|z_t, w)}{q(z_t|z_{t-1}, x)} - \underbrace{\log q(z_1|x)} + \log p(x|z_1, w) \right]$$


These two terms are constants with respect to w .

$\mathbb{E}_q[\log p(x|z_1, w)]$ corresponds to the **reconstruction term** from the VAE.

A Monte Carlo approximation:

$$\mathbb{E}_q[\log p(x|z_1, w)] \approx \sum_{l=1}^L \log p(x|z_1^{(l)}, w) \quad \text{where} \quad z_1^{(l)} \sim N(z_1 | \sqrt{1 - \beta_1} x, \beta_1 I).$$

Rewriting the ELBO

Now we will rewrite the second term to contain the **KL-divergence**.

From the Bayes rule: $q(z_t|z_{t-1}, x) = \frac{q(z_{t-1}|z_t, x)q(z_t|x)}{q(z_{t-1}|x)}$

$$\log \frac{p(z_{t-1}|z_t, w)}{q(z_t|z_{t-1}, x)} = \log \frac{p(z_{t-1}|z_t, w)}{q(z_{t-1}|z_t, x)} + \log \frac{q(z_{t-1}|x)}{q(z_t|x)}$$

This term is a constant with respect to w .

We then obtain

$$L(w) = \mathbb{E}_q \left[\sum_{t=2}^T \log \frac{p(z_{t-1}|z_t, w)}{q(z_{t-1}|z_t, x)} + \log p(x|z_1, w) \right].$$

Rewriting the ELBO

After marginalizing out unnecessary latent variables, we obtain

$$L(w) = \underbrace{\int q(z_1|x) \log p(x|z_1, w) dz_1}_{\text{Reconstruction term}} - \underbrace{\sum_{t=2}^T \int KL(q(z_{t-1}|z_t, x) || p(z_{t-1}|z_t, w)) q(z_t|x) dz_t}_{\text{Consistency terms}}.$$

Is this expression correct?
Prove it as an exercise 😊

The ELBO became very similar to that of VAE, but we have **multiple** encoder and decoder stages.

The **reconstruction term** rewards high probability for the observed data samples.
We use the Monte Carlo approximation for this term.

The **consistency terms** are between pairs of Gaussians! → We have **closed form** expressions!

Rewriting the ELBO

$$KL(q(z_{t-1}|z_t, x) || p(z_{t-1}|z_t, w)) = \underbrace{\frac{1}{2\beta_t} \|m_t(x, z_t) - \mu(z_t, w, t)\|^2}_{\text{Maximizing ELBO} \rightarrow \text{Minimizing the squared error}} + \underbrace{const.}_{\text{Other terms independent of } w \text{ are absorbed here.}}$$

$$p(z_{t-1}|z_t, w) = N(z_{t-1} | \mu(z_t, w, t), \beta_t I)$$

$$q(z_{t-1}|z_t, x) = N(z_{t-1} | m_t(x, z_t), \sigma_t^2 I)$$

$$\text{where } m_t(x, z_t) = \frac{(1 - \alpha_{t-1})\sqrt{1 - \beta_t}z_t + \sqrt{\alpha_{t-1}}\beta_t x}{1 - \alpha_t} \quad \text{and} \quad \sigma_t^2 = \frac{\beta_t(1 - \alpha_{t-1})}{1 - \alpha_t}.$$

For the expectation with respect to $q(z_t|x)$ for $\mathbb{E}_{q(z_t|x)} [KL(q(z_{t-1}|z_t, x) || p(z_{t-1}|z_t, w))]$, utilize the **Monte Carlo** approximation using $z_t \sim q(z_t|x) = N(z_t | \sqrt{\alpha_t}x, (1 - \alpha_t)I)$.

Predicting the Noise

One modification for higher quality results:

Instead of predicting **denoised image**, e.g., $\mu(z_t, w, t)$, at each time step, predict the **total noise** ϵ_t added to the original image to create the noised image at that step (Ho et al. 2020).

Noised image:
$$z_t = \sqrt{\alpha_t}x + \sqrt{1 - \alpha_t}\epsilon_t \quad \longrightarrow \quad x = \frac{1}{\sqrt{\alpha_t}}z_t - \frac{\sqrt{1 - \alpha_t}}{\sqrt{\alpha_t}}\epsilon_t$$

Rewrite the mean of $q(z_{t-1}|z_t, x)$ in terms of x and ϵ_t :

$$m_t(x, z_t) = \frac{(1 - \alpha_{t-1})\sqrt{1 - \beta_t}z_t + \sqrt{\alpha_{t-1}}\beta_t x}{1 - \alpha_t} \quad \longrightarrow \quad m_t(x, z_t) = \frac{1}{\sqrt{1 - \beta_t}} \left(z_t - \frac{\beta_t}{\sqrt{1 - \alpha_t}} \epsilon_t \right)$$

$$\alpha_t = \prod_{\tau=1}^t (1 - \beta_\tau)$$

Is this expression correct? Prove it as an exercise 😊

Introduce a model $g(z_t, w, t)$ to predict the total noise that was added to x to generate z_t :

$$\mu(z_t, w, t) = \frac{1}{\sqrt{1 - \beta_t}} \left(z_t - \frac{\beta_t}{\sqrt{1 - \alpha_t}} g(z_t, w, t) \right).$$

Predicting the Noise

Each term in the **consistency term** becomes as follows:

$$\begin{aligned} & KL(q(z_{t-1}|z_t, x) || p(z_{t-1}|z_t, w)) \\ &= \frac{1}{2\beta_t} \|m_t(x, z_t) - \mu(z_t, w, t)\|^2 + \text{const.} \\ &= \frac{\beta_t}{2(1 - \alpha_t)(1 - \beta_t)} \|g(z_t, w, t) - \epsilon_t\|^2 + \text{const.} \\ &= \frac{\beta_t}{2(1 - \alpha_t)(1 - \beta_t)} \left\| g\left(\sqrt{\alpha_t}x + \sqrt{1 - \alpha_t}\epsilon_t, w, t\right) - \epsilon_t \right\|^2 + \text{const.} \end{aligned}$$

Predicting the Noise

The **reconstruction term** can be approximated as follows:

$$\int q(z_1|x) \log p(x|z_1, w) dz_1 \approx \sum_{l=1}^L \log p(x|z_1^{(l)}, w) \quad \text{where} \quad z_1 = \sqrt{1 - \beta_1}x + \sqrt{\beta_1}\epsilon_1.$$

$$p(x|z_1, w) = N(x|\mu(z_1, w, 1), \beta_t I)$$

➡ $\log p(x|z_1, w) = -\frac{1}{2\beta_1} \|x - \mu(z_1, w, 1)\|^2 + \text{const.} \quad \alpha_1 = 1 - \beta_1$

➡ $\log p(x|z_1, w) = -\frac{1}{2(1 - \beta_1)} \|g(z_1, w, 1) - \epsilon_1\|^2 + \text{const.}$

This corresponds to the same form of the KL-divergence term with $t = 1$.

→ Reconstruction and consistency terms can be combined.

The Final Objective

$$L(w) = - \sum_{t=1}^T \left\| g \left(\sqrt{\alpha_t} x + \sqrt{1 - \alpha_t} \epsilon_t, w, t \right) - \epsilon_t \right\|^2$$

It is found that **omitting** the coefficients $\frac{\beta_t}{2(1-\alpha_t)(1-\beta_t)}$ can lead to a better performance (Ho et al. 2020). \rightarrow Objective at all time steps have **equal weighting**.

A simple interpretation: for a time step t and for a data point x , we **sample a noise** ϵ_t and use this to make the noisy latent vector z_t for that step. The **loss function** is the **squared error** between the **predicted noise** using $g(\cdot, \cdot, \cdot)$ and the **actual noise**. Note that $g(\cdot, \cdot, \cdot)$ is predicting the total noise added to x (not the incremental noise added in step t).

The optimization is performed using **stochastic gradient descent** (at each iteration, we randomly sample x and t to compute the gradient for the update).

Training Algorithm

Algorithm 20.1: Training a denoising diffusion probabilistic model

Input: Training data $\mathcal{D} = \{\mathbf{x}_n\}$

Noise schedule $\{\beta_1, \dots, \beta_T\}$

Output: Network parameters \mathbf{w}

for $t \in \{1, \dots, T\}$ **do**

$\alpha_t \leftarrow \prod_{\tau=1}^t (1 - \beta_\tau)$ // Calculate alphas from betas

end for

repeat

$\mathbf{x} \sim \mathcal{D}$ // Sample a data point

$t \sim \{1, \dots, T\}$ // Sample a point along the Markov chain

$\epsilon \sim \mathcal{N}(\epsilon | \mathbf{0}, \mathbf{I})$ // Sample a noise vector

$\mathbf{z}_t \leftarrow \sqrt{\alpha_t} \mathbf{x} + \sqrt{1 - \alpha_t} \epsilon$ // Evaluate noisy latent variable

$\mathcal{L}(\mathbf{w}) \leftarrow \|\mathbf{g}(\mathbf{z}_t, \mathbf{w}, t) - \epsilon\|^2$ // Compute loss term

 Take optimizer step

until converged

return \mathbf{w}

Generating New Samples

We can generate new samples in the data space by first **sampling from the Gaussian** $p(z_T)$ and then **denoising successively** through each step.

Given a denoised sample z_t at step t , we generate a sample z_{t-1} as follows:

1. Evaluate the output of $g(z_t, w, t)$.
2. Evaluate $\mu(z_t, w, t) = \frac{1}{\sqrt{1-\beta_t}} \left(z_t - \frac{\beta_t}{\sqrt{1-\alpha_t}} g(z_t, w, t) \right)$.
3. Generate a sample z_{t-1} from $p(z_{t-1}|z_t, w) = N(z_{t-1}|\mu(z_t, w, t), \beta_t I)$ by adding noise scaled by the variance, i.e., $z_{t-1} = \mu(z_t, w, t) + \sqrt{\beta_t}\epsilon$, where $\epsilon \sim N(0, I)$. (We do not add noise at the final step, i.e., when $t = 1$.)

Note: Even though $g(z_t, w, t)$ predicts the total noise ϵ_t added to x to obtain z_t , in the sampling step, we subtract off **only a fraction** $\frac{\beta_t}{\sqrt{1-\alpha_t}}$ of this noise and **then add additional noise with variance** β_t to generate z_{t-1} .

Sampling Algorithm

Algorithm 20.2: Sampling from a denoising diffusion probabilistic model

Input: Trained denoising network $\mathbf{g}(\mathbf{z}, \mathbf{w}, t)$

Noise schedule $\{\beta_1, \dots, \beta_T\}$

Output: Sample vector \mathbf{x} in data space

$\mathbf{z}_T \sim \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I})$ // Sample from final latent space

for $t \in T, \dots, 2$ **do**

$\alpha_t \leftarrow \prod_{\tau=1}^t (1 - \beta_\tau)$ // Calculate alpha

 // Evaluate network output

$\boldsymbol{\mu}(\mathbf{z}_t, \mathbf{w}, t) \leftarrow \frac{1}{\sqrt{1-\beta_t}} \left\{ \mathbf{z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}} \mathbf{g}(\mathbf{z}_t, \mathbf{w}, t) \right\}$

$\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}|\mathbf{0}, \mathbf{I})$ // Sample a noise vector

$\mathbf{z}_{t-1} \leftarrow \boldsymbol{\mu}(\mathbf{z}_t, \mathbf{w}, t) + \sqrt{\beta_t} \boldsymbol{\epsilon}$ // Add scaled noise

end for

$\mathbf{x} = \frac{1}{\sqrt{1-\beta_1}} \left\{ \mathbf{z}_1 - \frac{\beta_1}{\sqrt{1-\alpha_1}} \mathbf{g}(\mathbf{z}_1, \mathbf{w}, t) \right\}$ // Final denoising step

return \mathbf{x}

Some Drawbacks in Sampling

Diffusion models require **multiple sequential inference passes** in the **data space** through the trained network, which can be **computationally expensive**.

Many extensions to **reduce the computational cost** (e.g., reducing the required sampling steps or high-dimensionality) exist.

- Convert the discrete denoising process to a **differential equation over continuous time** and then apply alternative **efficient discretization methods** to reduce the number of function evaluations.
- **Denoising diffusion implicit models (DDIM):** Relax the Markovian assumption $q(x_t|x_{0:t-1}) = q(x_t|x_{t-1})$ on the noise process while using the same objective function for training (Song et al. 2020).
- **Latent diffusion models:** Train an **autoencoder** on noise-free images to obtain their low dimensional representations. Then train a **diffusion model** within the **low-dimensional latent space**. Finally, the denoised representation in the latent space is mapped into the data space using the decoder.

Guided Diffusion

Q. What if we want to **sample from a conditional distribution** $p(x|c)$?

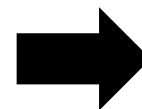
Here c could be a **class label** or a **textual description of the desired content in an image**.

This will also be a basis for other applications such as image super-resolution, image inpainting, video generation, and so on.

Treat c as an additional input and train $g(z_t, w, t, c)$?

The model can give insufficient weight to c .

We need a way to control how much weight is given to the **conditioning information** and to trade off against **sample diversity**.



We introduce a **guidance** to the diffusion model!

Classifier Guidance

It is shown that $g(z_t, w, t)$ is related to the density at t (Song and Ermon 2019): $g(z_t, w, t) \approx -\sqrt{1 - \alpha_t} \cdot \frac{\partial}{\partial z_t} \log q(z_t|x)$.

Suppose we have a **trained classifier** $p(c|x)$. Use the Bayes theorem and consider the gradient of $\log p$ with respect to x , called the **score function**:

$$\begin{aligned} \frac{\partial}{\partial x} \log p(x|c) &= \frac{\partial}{\partial x} \log \left(\frac{p(c|x) p(x)}{p(c)} \right) \\ &= \underbrace{\frac{\partial}{\partial x} \log p(x)}_{\text{red}} + \underbrace{\frac{\partial}{\partial x} \log p(c|x)}_{\text{green}} \end{aligned}$$

(Note that $\frac{\partial}{\partial x} \log p(c) = 0$.)

The first term: Usual unconditional score function

The second term: Direction that maximizes $p(c|x)$ under the classification model

By using the guidance scale λ :

$$\text{score}(x, c, \lambda) = \frac{\partial}{\partial x} \log p(x) + \lambda \frac{\partial}{\partial x} \log p(c|x)$$

$\lambda = 0$: the score of unconditional PDF

$\lambda = 1$: the score of conditional PDF

$\lambda = 10$: strongly respect the conditional label, but diversity can be low (easy-to-classify samples will be preferred)

Drawbacks: we need a **separate classifier**, and this should be able to classify examples with varying degrees of noise...

Classifier-Free Guidance

$$\begin{aligned} \text{score}(x, c, \lambda) &= \frac{\partial}{\partial x} \log p(x) + \lambda \frac{\partial}{\partial x} \log p(c|x) \\ \frac{\partial}{\partial x} \log p(x|c) &= \frac{\partial}{\partial x} \log p(x) + \frac{\partial}{\partial x} \log p(c|x) \end{aligned} \quad \rightarrow \quad \begin{aligned} \text{score}(x, c, \lambda) &= \lambda \frac{\partial}{\partial x} \log p(x|c) + (1 - \lambda) \frac{\partial}{\partial x} \log p(x) \end{aligned}$$

We can avoid training separate models for $p(x|c)$ and $p(x)$ by **training a single conditional model** with $p(x)$ is represented by $p(x|c = 0)$. (During training, we randomly set $c = 0$ with some probability (10~20%).)

It is known that **classifier-free guidance** gives much **higher quality** results than **classifier guidance**.

A possible reason: $p(c|x)$ can ignore most of the input as long as it makes a good prediction of c . **Classifier-free guidance** is based on $p(x|c)$, which must assign a high probability for all aspects of x .

Classifier-Free Guidance

Image
generation
under the
text
guidance.



Figure 20.7 Illustration of classifier-free guidance of diffusion models, generated from a model called GLIDE using the conditioning text *A stained glass window of a panda eating bamboo*. Examples on the left were generated with $\lambda = 0$ (no guidance, just the plain conditional model) whereas examples on the right were generated with $\lambda = 3$. [From Nichol *et al.* (2021) with permission.]

Classifier-Free Guidance

Image
super-
resolution.

Figure 20.8 Two examples of low-resolution images along with associated samples of corresponding high-resolution images generated by a diffusion model. The top row shows a 16×16 input image and the corresponding 128×128 output image along with the original image from which the input image was generated. The bottom row shows a 64×64 input image with a 256×256 output image, again with the original image for comparison. [From Saharia, Ho, *et al.* (2021) with permission.]

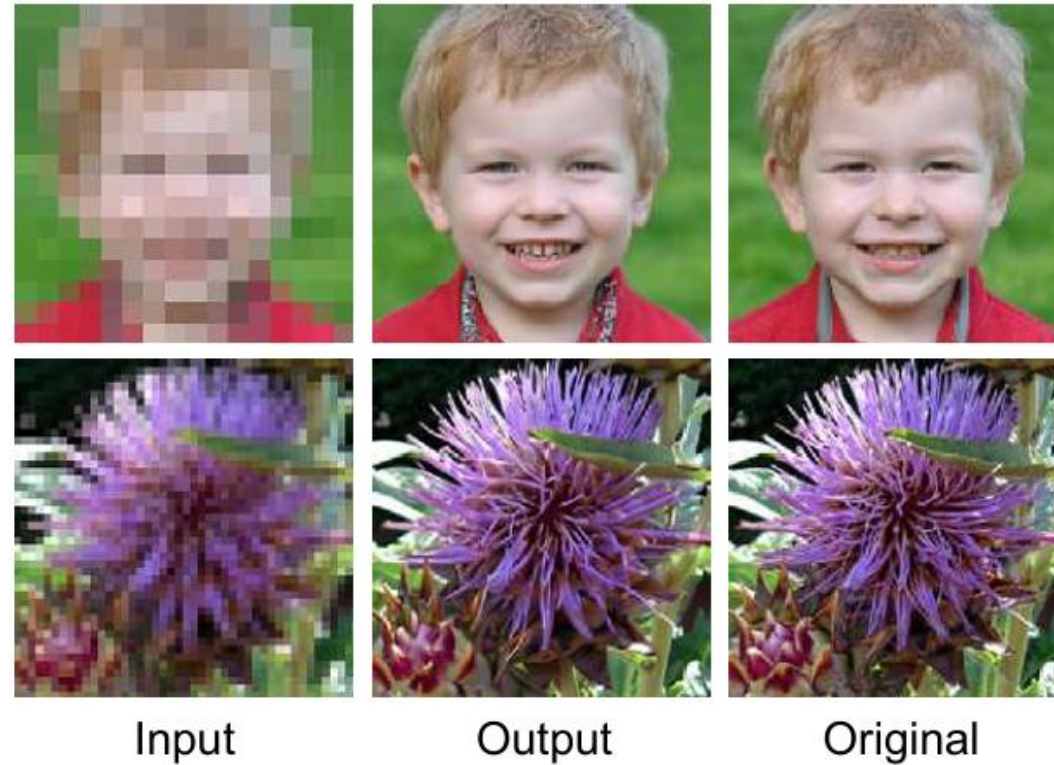


Image
inpainting.

Figure 20.9 Example of inpainting showing the original image on the left, an image with sections removed in the middle, and the image with inpainting on the right. [From Saharia, Chan, Chang, *et al.* (2021) with permission.]



Summary

- Diffusion
- Denoising Diffusion Probabilistic Models (DDPM)
 - Forward encoder
 - Reverse decoder
- Training DDPMs
 - Minimizing the evidence Lowerbound (ELBO) reduces to predicting the noise.
- Sampling new data from DDPMs
- Guided diffusion
 - Classifier guidance
 - Classifier-free guidance