Deep Generative Models

Chapter 6: Generation by Diffusion Process

Ali Bereyhi

`ali.bereyhi@utoronto.ca`

Department of Electrical and Computer Engineering

University of Toronto

Summer 2025

Recap: *Estimating from Noisy Observation*

Before we start with this section: *let us start with a simple problem*

Say we have data x and we see a noisy version of it as

$$z = x + \sigma\varepsilon$$

for noise process $\varepsilon \sim \mathcal{N}^0$

We intend to find the optimal function that

takes z and denoises it to an estimate of x

Bayesian Optimal Estimator

Let's think scalar

$$\hat{x}(z) = \operatorname*{argmin}_{u} \mathbb{E}\left\{ (u - x)^2 \,|z \right\}$$

To find the optimizer, we can write

$$\frac{\mathrm{d}}{\mathrm{d}u} \mathbb{E}\left\{ (u - x)^2 \,|z \right\} = 0$$

This concludes

$$\mathbb{E}\left\{ (u - x) \,|z \right\} = 0 \rightsquigarrow u^\star = \mathbb{E}\left\{ x|z \right\}$$

Bayes Optimal Denoiser

The optimal estimate in the Bayesian sense is

$$\hat{x}(z) = \mathbb{E}\left\{ x|z \right\}$$

Bayesian Denoiser

+ *Can we estimate the Bayes optimal denoiser by sampling?*
– Not really!

To estimate Bayes optimal denoiser by sampling: we need to

- *sample many data samples x whose noisy version is*

$$\boxed{\text{exactly } z}$$

- *use these samples to compute the estimator as*

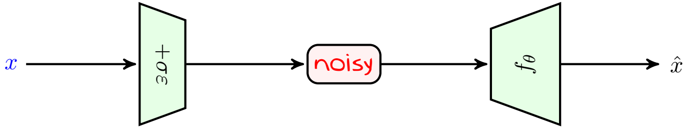
$$\hat{x}(z) = \mathbb{E}\{x|z\}$$

Complexity of Bayes Optimal Denoiser

To estimate the Bayes optimal denoiser, we need to collect exponentially large data samples, such that we have enough samples for each z!

Computational Denoising: *Decoding*

We can though train a computational denoiser



We train this model on a dataset using risk

$$
\begin{aligned}
R(\theta) &= \mathbb{E}_{x}\left\{\|\hat{x} - x\|^2\right\} \\
&= \mathbb{E}_{x}\left\{\|f_\theta(x + \sigma\varepsilon) - x\|^2\right\}
\end{aligned}
$$

This is equivalent to autoencoding with noisy encoder and decoder f_θ

Diffusion Process: *General SDE*

General Diffusion

A generic diffusion process is given by a stochastic differential equation (SDE)

$$\mathrm{d}\vec{x}(t) = f(\vec{x}(t), t)\,\mathrm{d}t + g(t)\,\mathrm{d}B(t)$$

which describes time evolution of location $\vec{x}(t)$ for a particle impacted by the Brownian motion process

+ *Why we write $\vec{x}(t)$?!*
– To indicate that we are doing forward in time

Note that the Langevin equation was a special case of this SDE

$$\mathrm{d}\vec{x}(t) = \nabla_x \log P(\vec{x}(t))\,\mathrm{d}t + \sqrt{2}\mathrm{d}B(t)$$

Diffusion Process: *Forward Process*

We use this SDE as follows: *if we shift Δt at time t, we approximately have*

$$\Delta \vec{x}(t) \approx f(\vec{x}(t), t) \Delta t + g(t) \Delta B(t)$$

Now, say we start at $t_0 = 0$ and move Δt step by step then

$$\begin{aligned} \vec{x}^{(i+1)} &\approx \vec{x}^{(i)} + \Delta \vec{x}(t_i) \\ &\approx \vec{x}^{(i)} + f\left(\vec{x}^{(i)}, t_i\right) \Delta t + g(t_i)\left[B(t_i + \Delta t) - B(t_i)\right] \\ &\approx \vec{x}^{(i)} + f^{(i)}\left(\vec{x}^{(i)}\right) \Delta t + g^{(i)} \sqrt{\Delta t} \varepsilon_i \end{aligned}$$

using the following notions and simplifications

- *we define $f^{(i)}\left(\vec{x}^{(i)}\right) = f\left(\vec{x}^{(i)}, t_i\right)$ and $g^{(i)} = g(t_i)$*
- *we use the fact that $B(t_i + \Delta t) - B(t_i) \sim \mathcal{N}(0, \Delta t)$*

Diffusion Process: *Forward Process*

+ *What is happening in this equation?*
– We are simulating the diffusion process

This is the forward process

$$\vec{x}^{(i+1)} = \vec{x}^{(i)} + f^{(i)}\left(\vec{x}^{(i)}\right)\Delta t + g^{(i)}\sqrt{\Delta t}\varepsilon_i$$

and we can look at it as a Markov chain

$$\vec{x}^{(0)} \sim P_0(x) \rightarrow \vec{x}^{(1)} \sim P_1(x) \rightarrow \vec{x}^{(2)} \sim P_2(x) \rightarrow \cdots \vec{x}^{(I)} \sim P_I(x)$$

where $P_i(x)$ is roughly describing the distribution of location at time

$$t = i\Delta t$$

Reverse Diffusion Process: *Reverse SDE*

B. Anderson in 1982 showed that *we can reverse this process in time*

Reverse Diffusion

The reverse diffusion process is given by the following SDE

$$\mathrm{d}\overleftarrow{x}(s) = \left[f\left(\overleftarrow{x}(s), s\right) - g(s)^2 \nabla_x \log P\left(\overleftarrow{x}(s)\right) \right] \mathrm{d}s + g(s)\, \mathrm{d}B(s)$$

where $s = T - t$ *is the reverse time started as* T

Reverse Dynamics

+ *What is really reverse in time?*
– We can move statistically backward in time

We can again approximate this equation by a reverse Markov chain

$$\overleftarrow{x}^{(i+1)} = \overleftarrow{x}^{(i)} + \left[\overleftarrow{f}^{(i)} \left(\overleftarrow{x}^{(i)} \right) - \left(\overleftarrow{g}^{(i)} \right)^2 \nabla_x \log P_{I-i} \left(\overleftarrow{x}^{(i)} \right) \right] \Delta t + \overleftarrow{g}^{(i)} \sqrt{\Delta t} \varepsilon_i$$

Now, we start by a sample $\overleftarrow{x}^{(0)} \sim P_I(x)$ *and proceed as*

$$\overleftarrow{x}^{(0)} \sim P_I(x) \to \overleftarrow{x}^{(1)} \sim P_{I-1}(x) \to \overleftarrow{x}^{(2)} \sim P_{I-2}(x) \to \cdots \overleftarrow{x}^{(I)} \sim P_0(x)$$

where $P_i(x)$ *is again describing the distribution of location at time*

$$t = i\Delta t$$

Diffusing Back and Forth

Let's do some simplification: *we compactly show the forward diffusion as*

$$\vec{x}^{(i+1)} \sim \vec{Q}_i\left(\vec{x}^{(i+1)} | \vec{x}^{(i)}\right)$$

and the reverse diffusion as

$$\overleftarrow{x}^{(i+1)} \sim \overleftarrow{Q}_i\left(\overleftarrow{x}^{(i+1)} | \overleftarrow{x}^{(i)}\right)$$

Equivalency in Opposite Direction

If we know a forward diffusion $\vec{Q}_i$ that takes us

from $x^{(0)} \sim P^{(0)}(x)$ to $x^{(I)} \sim P^{(I)}(x)$

Then, we can build the reverse diffusion $\overleftarrow{Q}_i$ which can take us

back from $x^{(I)} \sim P^{(I)}(x)$ to $x^{(0)} \sim P^{(0)}(x)$

Noising Diffusion: *Data to Latent*

+ *Why should this property be helpful?!*
- We know how to get from a data sample to noise

Consider the following forward diffusion process

$$\vec{x}^{(i+1)} = \vec{x}^{(i)} - \underbrace{\beta^{(i)}\vec{x}^{(i)}}_{f(\vec{x}(t),t)} \Delta t + \underbrace{\sqrt{2\beta^{(i)}\left(1 - \frac{\beta^{(i)}\Delta t}{2}\right)}}_{g(t)} \sqrt{\Delta t}\varepsilon_i$$

If we define $\alpha^{(i)} = 1 - \beta^{(i)}\Delta t$*; then, it simplifies to*

$$\vec{x}^{(i+1)} = \alpha^{(i)}\vec{x}^{(i)} + \sqrt{1 - {\alpha^{(i)}}^2}\varepsilon_i$$

and we use the notation $\bar{\alpha}^{(i)} = \sqrt{1 - {\alpha^{(i)}}^2}$ *for simplicity*

Noising Diffusion: *Data to Latent*

The following Markov chain

$$\vec{x}^{(i+1)} = \alpha^{(i)} \vec{x}^{(i)} + \bar{\alpha}^{(i)} \varepsilon_i$$

describes an Ornstein-Uhlenbeck (OU) process: starting with a centered and normalized sample of any distribution

- *the OU process computes always a unit-variance sample for next time*
- *as it proceeds for large I, we get*

$$\vec{x}^{(I)} \sim \mathcal{N}(0, 1)$$

Nosing Process

We know a diffusion process that turns a data sample $\vec{x}^{(0)}$ into Gaussian latent

Denoising Diffusion: *Latent to Data*

Since we know the forward diffusion: *we can*

build the reverse diffusion to get back from latent to data

This was the forward process

$$\vec{x}^{(i+1)} = \vec{x}^{(i)} - \beta^{(i)} \vec{x}^{(i)} \Delta t + \sqrt{2\beta^{(i)} \left(1 - \frac{\beta^{(i)} \Delta t}{2}\right)} \sqrt{\Delta t} \varepsilon_i$$

We need to change the drift and diffusion coefficient to

$$\overleftarrow{f}^{(i)} \left(\overleftarrow{x}^{(i)}\right) = -\beta^{(I-i)} \overleftarrow{x}^{(i)} - 2\beta^{(I-i)} \left(1 - \frac{\beta^{(I-i)} \Delta t}{2}\right) \nabla_x \log P_{I-i} \left(\overleftarrow{x}^{(i)}\right)$$

$$\overleftarrow{g}^{(i)} \left(\overleftarrow{x}^{(i)}\right) = \sqrt{2\beta^{(I-i)} \left(1 - \frac{\beta^{(I-i)} \Delta t}{2}\right)}$$

Denoising Diffusion: *Latent to Data*

We can then denoise the latent into a data sample by

1. *Sample a Gaussian* $\overleftarrow{x}^{(0)} \sim \mathcal{N}(0,1)$
2. *Proceed in reverse time as*

$$\overleftarrow{x}^{(i+1)} = \vec{x}^{(i)} + \overleftarrow{f}^{(i)}\left(\overleftarrow{x}^{(i)}\right)\Delta t + \overleftarrow{g}^{(i)}\left(\overleftarrow{x}^{(i)}\right)\sqrt{\Delta t}\varepsilon_i$$

3. *Take* $\overleftarrow{x}^{(I)} \sim P(x)$ *as a data sample*

+ *Do we have everything that we need?!*
– Let's break it down

Required Scores for Denoising Diffusion

We know the diffusion process

$$\vec{x}^{(0)} \sim P_{\text{data}} \to \vec{x}^{(1)} \sim P_1 \to \vec{x}^{(2)} \sim P_2 \to \cdots \to \vec{x}^{(I)} \sim \mathcal{N}(0,1)$$

Just set a data sample at $\vec{x}^{(0)} \sim P_{\text{data}}$ *and diffuse as*

$$\vec{x}^{(i+1)} = \alpha^{(i)} \vec{x}^{(i)} + \bar{\alpha}^{(i)} \varepsilon_i$$

We can also sample latent $\overleftarrow{x}^{(0)} \sim \mathcal{N}(0,1)$,*but to diffuse reversely as*

$$\overleftarrow{x}^{(0)} \sim \mathcal{N}(0,1) \to \overleftarrow{x}^{(1)} \sim P_{I-1} \to \cdots \to \overleftarrow{x}^{(I-1)} \sim P_1 \to \overleftarrow{x}^{(I)} \sim P_{\text{data}}$$

we need to know the following score functions

$$\nabla_x \log \mathcal{N}^0(x), \nabla_x \log P_{I-1}(x), \ldots, \nabla_x \log P_1(x)$$

Score of Noisy Output

+ *Back to square one! We need score function of the data!*
- No! We need *all scores but data score!*

We need to learn these scores

$$\nabla_x \log P_{I-1}(x), \ldots, \nabla_x \log P_1(x)$$

1. $\nabla_x \log P_1(x)$ *is the score of*

$$\vec{x}^{(1)} = \alpha^{(0)} \underbrace{x}_{\text{data sample}} + \bar{\alpha}^{(0)} \varepsilon_0$$

2. $\nabla_x \log P_2(x)$ *is the score of*

$$\begin{aligned}\vec{x}^{(2)} = \alpha^{(1)} \vec{x}^{(1)} + \bar{\alpha}^{(1)} \varepsilon_1 &= \alpha^{(1)} \left(\alpha^{(0)} x + \bar{\alpha}^{(0)} \varepsilon_0 \right) + \bar{\alpha}^{(1)} \varepsilon_1 \\ &= \alpha^{(1)} \alpha^{(0)} x + \left(\alpha^{(1)} \bar{\alpha}^{(0)} \varepsilon_0 + \bar{\alpha}^{(1)} \varepsilon_1 \right)\end{aligned}$$

Score of Noisy Output

In fact we need *the score function of a noisy version of* x, *i.e.,*

$$z = x + \sigma\varepsilon$$

for some σ *when* $\varepsilon \sim \mathcal{N}(0, 1)$

- + *How is it so different then?!*
- – *It turns out to be very different! Let's take a look*

Problem: *Score of Noisy Sample*

Let $x \sim P$ *and* $z = x + \sigma\varepsilon$; *find the score*

$$s(z) = \nabla_z \log P(z)$$

with $P(z)$ *being the marginal distribution of* z

Score of Noisy Output

Let's think about scalar x *and* z

↳ *everything extends the same to multi-dimensional case*

We can write

$$P(z) = \int P(z|x)\, P(x)\, \mathrm{d}x$$

We know that $P(z|x) \equiv \mathcal{N}(x, \sigma^2)$*: so, we could write*

$$P(z) = \frac{1}{\sqrt{2\pi\sigma^2}} \int \exp\left\{-\frac{(z-x)^2}{2\sigma^2}\right\} P(x)\, \mathrm{d}x$$

Score of Noisy Output

For score function we have

$$\nabla_z \log P(z) = \frac{1}{P(z)} \nabla_z P(z)$$

Considering the expression

$$P(z) = \frac{1}{\sqrt{2\pi\sigma^2}} \int \exp\left\{-\frac{(z-x)^2}{2\sigma^2}\right\} P(x)\, \mathrm{d}x$$

we can easily write

$$\begin{aligned} \nabla_z P(z) &= \frac{1}{\sqrt{2\pi\sigma^2}} \int \frac{(z-x)}{\sigma^2} \exp\left\{-\frac{(x-z)^2}{2\sigma^2}\right\} P(x)\, \mathrm{d}x \\ &= \int \frac{(x-z)}{\sigma^2} P(z|x) P(x)\, \mathrm{d}x \end{aligned}$$

Score of Noisy Output

So, we can write

$$\begin{aligned}
\nabla_z \log P(z) &= \frac{1}{P(z)} \int \frac{(x-z)}{\sigma^2} P(z|x) P(x)\, \mathrm{d}x \\
&= \frac{1}{P(z)} \int \frac{(x-z)}{\sigma^2} P(z,x) \mathrm{d}x \\
&= \frac{1}{P(z)} \int \frac{(x-z)}{\sigma^2} P(x|z) P(z) \mathrm{d}x \\
&= \int \frac{(x-z)}{\sigma^2} P(x|z) \mathrm{d}x = \mathbb{E}_{x \sim P_{|z}} \left\{ \frac{(x-z)}{\sigma^2} \right\}
\end{aligned}$$

This concludes that

$$s(z) = \frac{1}{\sigma^2} \mathbb{E}_x \{x - z | z\} = \frac{1}{\sigma^2} \left(\underbrace{\mathbb{E}_x \{x|z\}}_{\hat{x}(z)} - z \right) = \frac{\hat{x}(z) - z}{\sigma^2}$$

Score of Noisy Output

We could also say that

$$z = x + \sigma\varepsilon \rightsquigarrow \varepsilon = \frac{z - x}{\sigma}$$

So, we could also write

$$s(z) = \frac{1}{\sigma}\mathbb{E}_x\left\{\frac{x - z}{\sigma}\Big|z\right\} = -\frac{1}{\sigma}\mathbb{E}\left\{\varepsilon|z\right\}$$

Score of Noisy Output

Problem: *Score of Noisy Sample*

Let $x \sim P$ *and* $z = x + \sigma\varepsilon \rightsquigarrow$ *find the score* $s(z)$

The score at point z *is given by*

- *either the conditional noise average*

$$s(z) = -\frac{\mathbb{E}\{\varepsilon | z\}}{\sigma}$$

- *or the Bayes optimal denoiser as*

$$s(z) = \frac{\hat{x}(z) - z}{\sigma^2}$$

Computing Score by *Noise Averaging*

Say we need to find score $s(z)$ while we have access to a bunch of noise samples ε^j with the following property

↳ *All $z - \sigma\varepsilon$ end up with a data sample*

We can then estimate the score as

$$s(z) = -\frac{\mathbb{E}\{\varepsilon | z\}}{\sigma} \approx -\frac{\hat{\mathbb{E}}\{\varepsilon | z\}}{\sigma}$$

+ *But how could we a bunch of such noise samples?!*
- *We rely on the single sample that we have* ☺

> *In practice we estimate as*
>
> $$s(z) \approx -\frac{\varepsilon}{\sigma}$$

Computing Score by *Denoising*

Say we need to find score $s(z)$ while we have access to data sample x: we take x and sample $\varepsilon \sim \mathcal{N}^0$; then, we noise *the data sample as*

$$z = x + \sigma\varepsilon$$

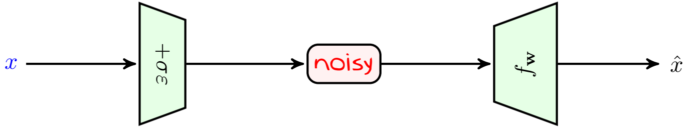
We can then use the denoiser $\hat{x}(z)$ to compute the score as

$$s(z) = \frac{\hat{x}(z) - z}{\sigma^2}$$

+ *But how could we get the denoiser $\hat{x}(z)$?!*
– *We can learn it!*

Computational Approach: *Decoding*

We saw before that *we can denoise by a computational model*



We just need to train this model by risk

$$\begin{aligned} R(\mathbf{w}) &= \mathbb{E}_x \left\{ \|\hat{x} - x\|^2 \right\} \\ &= \mathbb{E}_x \left\{ \| f_{\mathbf{w}}(x + \sigma\varepsilon) - x\|^2 \right\} \end{aligned}$$

Training Diffusion Score Model: *Direct Score Model*

We consider *a score model* $s_{\mathbf{w}} : \mathbb{R}^d \times \mathbb{R} \mapsto \mathbb{R}^d$ *which*

takes a noisy sample z *and noise standard deviation* $\sigma \rightsquigarrow$ *returns* $s(z)$

To train this model we make data batches as follows

For $j = 1, \ldots, n$

1. *Sample a data sample* x^j *and choose a value for* σ^j
2. *Noise data sample as* $z^j = x^j + \sigma^j \varepsilon^j$ *for some* $\varepsilon^j \sim \mathcal{N}^0$
3. *Set the data point and its label (score) to*

$$\text{input to } s_{\mathbf{w}} \equiv \left(z^j, \sigma^j\right) \qquad \text{output of } s_{\mathbf{w}} \equiv -\varepsilon^j / \sigma^j$$

We could train this network by minimizing the empirical loss

$$\hat{R}(\mathbf{w}) = \hat{\mathbb{E}}\left\{\|s_{\mathbf{w}}\left(z^j, \sigma^j\right) + \varepsilon^j/\sigma^j\|^2\right\}$$

Training Diffusion Score Model: *Denoising Model*

We consider *a denosing model* $f_\theta : \mathbb{R}^d \times \mathbb{R} \mapsto \mathbb{R}^d$ *which*

takes a noisy sample z *and noise standard deviation* $\sigma \rightsquigarrow$ *denoises to* $\hat{x}(z)$

To train this model we make data batches as follows

For $j = 1, \ldots, n$

1. *Sample a data sample* x^j *and choose a value for* σ^j
2. *Noise data sample as* $z^j = x^j + \sigma^j \varepsilon^j$ *for some* $\varepsilon^j \sim \mathcal{N}^0$
3. *Set the data point and its label (recovery) to*

$$\text{input to } f_{\mathbf{w}} \equiv \left(z^j, \sigma^j\right) \qquad \text{output of } f_{\mathbf{w}} \equiv x^j$$

We can use this denoiser to estimate the score as

$$s_{\mathbf{w}}(z, \sigma) = \frac{f_{\mathbf{w}}(z, \sigma) - z}{\sigma^2}$$

Sampling Diffusion Score Model: *Forward Diffusion*

We define a forward diffusion as

$$\vec{x}^{(i+1)} = \alpha^{(i)} \vec{x}^{(i)} + \bar{\alpha}^{(i)} \varepsilon_i$$

+ *How should we choose* $\alpha^{(i)}$*?!*
- Well! *For sure* $\bar{\alpha}^{(i)}$ *is small, since we defined* $\alpha^{(i)} = 1 - \beta^{(i)} \Delta t$

For this process we can say

$$\begin{aligned}
\vec{x}^{(1)} &= \alpha^{(0)} x + \bar{\alpha}^{(0)} \varepsilon_0 && = \theta^{(0)} x + \sigma^{(0)} \varepsilon^{(0)} \\
\vec{x}^{(2)} &= \alpha^{(1)} \alpha^{(0)} x + \left(\alpha^{(1)} \bar{\alpha}^{(0)} \varepsilon_0 + \bar{\alpha}^{(1)} \varepsilon_1 \right) && = \theta^{(1)} x + \sigma^{(1)} \varepsilon^{(1)} \\
&\vdots
\end{aligned}$$

with all $\varepsilon^{(i)} \sim \mathcal{N}^0$

Sampling Diffusion Score Model: *Reverse Diffusion*

From the forward diffusion: *we can build the reverse diffusion as*

$$\overleftarrow{x}^{(i+1)} = \left[\overleftarrow{f}^{(i)} \overleftarrow{x}^{(i)} - \overleftarrow{g}^{(i)2} s \left(\overleftarrow{x}^{(i)} \right) \right] + \overleftarrow{g}^{(i)} \varepsilon_i$$

Noting that $\overleftarrow{x}^{(i)} = \vec{x}^{(I-i)}$ *and*

$$\vec{x}^{(i)} = \theta^{(i-1)} x + \sigma^{(i-1)} \varepsilon^{(i-1)}$$

We re-write $\overleftarrow{x}^{(i)} = x + \hat{\sigma}^{(i-1)} \varepsilon^{(i-1)}$ *and use* $s_{\mathbf{w}}$ *to estimate the score as*

$$s \left(\overleftarrow{x}^{(i)} \right) \approx s_{\mathbf{w}} \left(\overleftarrow{x}^{(i)}, \hat{\sigma}^{(i-1)} \right)$$

We keep going backward till we get to

$$\overleftarrow{x}^{(I)} \sim P_0 \equiv P_{\text{data}}$$