Diffusion Models and Inverse Problems

Pramook Khungurn

March 19, 2025

This note is written as I read the survey paper “A Survey on Diffusion Models for Inverse Problems” by Dara et al. [DCL$^+$24].

1 Problem Setting

- A data item is denoted by $\mathbf{x} \in \mathbb{R}^n$.
- We are interested in a random variable $\mathbf{X}$ whose values are data items. The probability distribution of $\mathbf{X}$ is denoted by $p_{\mathbf{X}}$.
- Given a data item $\mathbf{x}$, we extract from it a **measurement** $\mathbf{y} \in \mathbb{R}^m$.
- We model the measurement process with a **measurement model**, which is also called a **forward model** or a **corruption model** in literature. That is, we establish a random variable $\mathbf{Y}$ whose values are measurements according to the following equation.

$$\mathbf{Y} = \mathcal{A}(\mathbf{X}) + \sigma_{\mathbf{y}}\mathbf{Z}$$

  where $\mathcal{A} : \mathbb{R}^n \to \mathbb{R}^m$ is a deterministic function and $\mathbf{Z} \sim \mathcal{N}(0, I)$. The function $\mathcal{A}$ is often called the **forward measurement operator** or just the **measurement operator**.
- In an inverse problem, we are given a measurement $\mathbf{y}$. The goal is to produce a data item $\mathbf{x}$ that could have produced $\mathbf{y}$ according to the measurement model.
- Examples of inverse problems. Here, $\sigma_{\mathbf{y}} = 0$ unless noted otherwise.
  - **Denoising.** $\mathcal{A}$ is the identitfy function, and $\sigma_{\mathbf{y}} > 0$. This results in the measurement $\mathbf{y}$ being $\mathbf{x}$ with some added noise.
  - **Inpainting.** $\mathcal{A}$ zeros out some components $\mathbf{x}$.
  - **Compressed sensing.** $\mathcal{A}(\mathbf{x}) = A\mathbf{x}$ where A is a matrix with entries sampled from a Gaussian random variable.
  - **Signal recovery from convolution.** $\mathcal{A}(\mathbf{x}) = \mathbf{x} * \mathbf{k}$ where $*$ denotes convolution. **Deblurring** and **super-resolution** are examples of signal recovery from convolution.

  Note that all of the examples above are *linear* in the sense that $\mathcal{A}(\mathbf{x}) = A\mathbf{x}$ for some matrix $A \in \mathbb{R}^{m\times n}$. Morevoer, $m = n$ in all of them. However, these constraints need not hold in general. Here is an example of a nonlinear inverse problem.

  - **Phase retrieval.** $\mathcal{A}(\mathbf{x}) = |\mathscr{F}(\mathbf{x})|$ where $\mathscr{F}$ is the discrete Fourier transform operator, and $|\cdot|$ denotes the norm of a complex number. Here, the measurement is the norms of the Fourier coefficients, which means that the phase information has been thrown away.

- **Decompression.** $\mathcal{A}$ is a non-linear function that compresses $\mathbf{x}$. For example, $\mathbf{x}$ can be an image, and $\mathcal{A}$ is the JPEG compression algorithm.

- One of the characteristic of the above problems are that perfect recovery is impossible.
  - In other words, these problems are *ill-posed*.
- As result, we almost always have say exactly what type of recovery we want.
  - **Posterior sampling.** We do not care about the exact $\mathbf{x}$, but we just want to sample from the **posterior** distribution $p_{\mathbf{X}}(\mathbf{x}|\mathbf{Y}=\mathbf{y})$.
    * For brevity, we shall write this as just $p(\mathbf{x}|\mathbf{y})$.
  - **Maximum a posterior (MAP) estimation.** We look for $\mathbf{x}$ that maximizes $p(\mathbf{x}|\mathbf{y})$.
  - **Minimum mean square error (MMSE) estimation.** We seek $\mathbf{x}^*$ that minimizes
  $$\mathcal{L}(\mathbf{x}^*) = E_{\mathbf{x}\sim p(\mathbf{x}|\mathbf{y})}[\|\mathbf{x}-\mathbf{x}^*\|^2].$$
  This results in the expectation $E[\mathbf{X}|\mathbf{Y}=\mathbf{y}]$ (or just simply $E[\mathbf{x}|\mathbf{y}]$).
- There are limitations to MAP and MMSE estimations [BM18, DM24].
  - They can suffer from "regression to the mean."
  - This means that the predicted $\mathbf{x}$ may lack important details and be outside the desire solution space.

  So, in this note, the focus is on posterior sampling.
- As the title of this note implies, we are interested in using diffusion models to sample from $p(\mathbf{x}|\mathbf{y})$.
- A direct way to do so is to just train a conditional diffusion model to do this job.
- Indeed, there are many conditional diffusion models for specific inverse problems such as:
  - super-resolution [LYC+21, SHC+21],
  - deblurring [WDT+21],
  - inpainting [SCC+22], and
  - image restoration [SCC+22, LGZ+23a, LGZ+23b].

  Even ControlNet [ZRA23] and similar approches can be thought of an instance of solving an inverse problem.
- However, these approches require training a new diffusion model fron scratch or fine-tuning existing ones. This can be quite expensive.
- Instead, we focus on the **unsupervised approach**, in which we take an existing diffusion model and use it to sample from $p(\mathbf{x}|\mathbf{y})$ without any training or fine-tuning.
- Let us first see how using a diffusion model can be useful. To see this we need to mathematically examine posterior sampling. According to Bayes' rule, we have that
  $$p(\mathbf{x}|\mathbf{y}) = p(\mathbf{X}=\mathbf{x}|\mathbf{Y}=\mathbf{y}) = \frac{p(\mathbf{Y}=\mathbf{y}|\mathbf{X}=\mathbf{x})p(\mathbf{X}=\mathbf{x})}{p(\mathbf{Y}=\mathbf{y})} = \frac{p_{\mathbf{Y}}(\mathbf{y}|\mathbf{x})p_{\mathbf{X}}(\mathbf{x})}{p_{\mathbf{Y}}(\mathbf{y})}.$$
  We generally do not care about $p_{\mathbf{Y}}(\mathbf{y})$ because $\mathbf{y}$ is already given to us. So, we usually just write
  $$p(\mathbf{x}|\mathbf{y}) \propto p(\mathbf{y}|\mathbf{x})p_{\mathbf{X}}(\mathbf{x}).$$
  Our task thus becomes sampling from the unnormalized distribution $p(\mathbf{y}|\mathbf{x})p_{\mathbf{X}}(\mathbf{x})$. The distribution $p_{\mathbf{X}}(\mathbf{x})$ is called the **prior** and $p(\mathbf{y}|\mathbf{x})$ is called the **likelihood**.

- A pre-trained diffusion model can be useful because we can use it as a prior $p_{\mathbf{X}}$. However, there are several difficulty in this direction.
- First, given a sample $\mathbf{x}$, a pre-trained diffusion model does not allow us evaluate $p_{\mathbf{X}}(\mathbf{x})$ directly.
- Second, we need to find a way to somehow approximate $p(\mathbf{y}|\mathbf{x})$ or something related to it with the diffusion model.

2 Background

2.1 Diffusion Processes

- Given a data distribution $p_{\mathbf{X}}$, we can construct a stochastic process $\{\mathbf{X}_t : 0 \leq t \leq T\}$ such that $\mathbf{X}_0 \sim p_{\mathbf{X}}$ and $\mathbf{X}_T$ is approximately distributed according to $\mathcal{N}(\mathbf{0}, \sigma_T^2 I)$ for some $\sigma_T > 0$. This is done through a stochastic differential equation (SDE):
$$\mathrm{d}\mathbf{X}_t = \mathbf{f}(\mathbf{X}_t, t)\,\mathrm{d}t + g(t)\,\mathrm{d}\mathbf{W}_t \tag{1}$$
where W_t is the standard Brownian motion in $\mathbb{R}^n$. The stochastic process we desire is the solution to the above SDE with the intial condition $\mathbf{X}_0 \sim p_{\mathbf{X}}$.
- Let us give names to functions inside the above SDE.
  - The function $\mathbf{f} : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$ is called the **drift coefficient**.
  - The function $g : \mathbb{R} \to \mathbb{R}$ is called the **diffusion coefficient**.
- We will be working with the distribution of $\mathbf{X}_t$, which is denoted by $p_{\mathbf{X}_t}$ under standard notation. However, for convenience and brevity, let us use p_t instead of $p_{\mathbf{X}_t}$. In other words,
$$p_t(\mathbf{x}_t) = p_{\mathbf{X}_t}(\mathbf{x}_t) = p(\mathbf{X}_t = \mathbf{x}_t).$$
- According to Anderson [And82], we can sample p_0 by solving the reverse SDE
$$\mathrm{d}\mathbf{X}_t = \Big(\mathbf{f}(\mathbf{X}_t, t) - g^2(t)\nabla_{\mathbf{X}_t}\log p_t(\mathbf{X}_t)\Big)\,\mathrm{d}t + g(t)\,\mathrm{d}\overline{\mathbf{W}}_t \tag{2}$$
from $t = T$ to $t = 0$, initializing $\mathbf{X}_T$ by sampling from p_T, an isotropic Gaussian distribution. Here, $\overline{\mathbf{W}}_t$ is the standard Brownnian motion that runs backwards in time.
- Song et al. observes that the following ODE,
$$\frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}_t, t) - \frac{g^2(t)}{2}\nabla_{\mathbf{x}_t}\log p_t(\mathbf{x}_t), \tag{3}$$
satisfies the same Fokker–Planck equation as the SDE (1) [SSDK+21]. Hence, we also have a deterministic sampling scheme to sample from $p_{\mathbf{X}}$.
- There are two variants of the SDE in Equation 1 that are widely used.
  - **Variance exploding SDE.**
    - We set $\mathbf{f}(\mathbf{X}_t, t) = \mathbf{0}$.
    - The SDE (1) then boils down to the following relationship between random variables.
$$\mathbf{X}_t = \mathbf{X}_0 + \sigma_t \mathbf{Z}$$
where $\mathbf{X}_0 \sim p_0 = p_{\mathbf{X}}$, $\mathbf{Z} \sim \mathcal{N}(\mathbf{0}, I)$, and
$$\sigma_t = \sqrt{\int_0^t g^2(u)\,\mathrm{d}u}.$$

    - A typical noise schedule is $\sigma_t = t$, which corresponds to $g(t) = \sqrt{2t}$, and $\mathbf{X}_t = \mathbf{X}_0 + t\mathbf{Z}$. This schedule is used in the famous EDM paper [KAAL22].
  - **Variance preversing SDE.**
    - We choose the drift and the diffusion coefficients so that the following relationship holds:

$$\mathbf{X}_t = \alpha_t \mathbf{X}_0 + \sigma_t \mathbf{Z}, \text{ and } \alpha_t^2 + \sigma_t^2 = 1.$$

    - Again, $\mathbf{X}_0 \sim p_0 = p_{\mathbf{X}}$ and $\mathbf{Z} \sim \mathcal{N}(\mathbf{0}, I)$.
    - The functions $\alpha_t : [0,T] \to \mathbb{R}$ and $\sigma_t : [0,T] \to \mathbb{R}$ are both called "noise schedules."
    - An example of a variance preserving SDE is the following Ornstein–Uhlenbeck process,

$$\mathrm{d}\mathbf{X}_t = -\mathbf{X}_t \,\mathrm{d}t + \sqrt{2}\,\mathrm{d}\mathbf{W}_t,$$

    which gives $\alpha_t = \exp(-t)$ and $\sigma_t = \sqrt{1 - \exp(-2t)}$.

- In general, we will be using an SDE of the form

$$\mathrm{d}\mathbf{X}_t = f(t)\mathbf{X}_t \,\mathrm{d}t + g(t)\,\mathrm{d}\mathbf{W}_t$$

  for some $f : \mathbb{R} \to \mathbb{R}$. This equation gives us the relation

$$\mathbf{X}_t = \alpha_t \mathbf{X}_0 + \sigma_t \mathbf{Z} \tag{4}$$

  where $\mathbf{X}_0 \sim p_0 = p_{\mathbf{X}}$ and $\mathbf{Z} \sim \mathcal{N}(\mathbf{0}, I)$. The noise schedules α_t and σ_t are related to $f(t)$ and $g(t)$ as follows [KAAL22]:

$$\begin{aligned}
\alpha_t &= \exp\left(\int_0^t f(u)\,\mathrm{d}u \right), \\
\sigma_t &= \sqrt{\int_0^t \frac{g^2(u)}{\alpha_u^2}\,\mathrm{d}u}, \\
f(t) &= \frac{\dot{\alpha}_t}{\alpha_t}, \\
g(t) &= \alpha_t \sqrt{2\sigma_t \dot{\sigma}_t}.
\end{aligned}$$

  Here, $\dot{\alpha}_t$ and $\dot{\sigma}_t$ are derivatives of α_t and σ_t with respect to time.

2.2 Tweedie's Formula and Diffusion Model Training

- **Theorem 1 (Tweedie's formula.).** *Let $\mathbf{X}$ and $\mathbf{Y}$ be random variables such that $\mathbf{Y} = \mathbf{X} + \sigma\mathbf{Z}$ where $\mathbf{Z} \sim \mathcal{N}(\mathbf{0}, I)$. It follows that*

$$E[\mathbf{X}|\mathbf{Y} = \mathbf{y}] = \mathbf{y} + \sigma^2 \nabla_{\mathbf{y}} \log p_{\mathbf{Y}}(\mathbf{y}).$$

- The expression $\nabla_{\mathbf{y}} \log p_{\mathbf{Y}}(\mathbf{y})$ is called the **score** of $p_{\mathbf{Y}}$.

- Applying Tweedie's formula to Equation (4), we have that

$$E[\alpha_t \mathbf{X}_0 | \mathbf{X}_t = \mathbf{x}_t] = \mathbf{x}_t + \sigma_t^2 \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t).$$

  In other words,

$$E[\mathbf{X}_0 | \mathbf{X}_t = \mathbf{x}_t] = \frac{\mathbf{x}_t + \sigma_t^2 \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)}{\alpha_t}.$$

Or, more concisely,

$$E[\mathbf{x}_0|\mathbf{x}_t] = \frac{\mathbf{x}_t + \sigma_t^2 \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)}{\alpha_t}. \tag{5}$$

This gives

$$\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) = \frac{\alpha_t E[\mathbf{x}_0|\mathbf{x}_t] - \mathbf{x}_t}{\sigma_t^2}. \tag{6}$$

- A diffusion model is a neural network $\mathbf{h}_{\boldsymbol{\theta}}$ with parameters $\boldsymbol{\theta}$ such that $\mathbf{h}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)$ approximates $E[\mathbf{x}_0|\mathbf{x}_t]$. It can be trained with the following **denoising score matching** objective

$$\mathcal{L}_{\mathrm{DSM}}(\boldsymbol{\theta}) = E_{t,\mathbf{x}_0 \sim p_{\mathbf{X}}, \mathbf{z} \sim \mathcal{N}(\mathbf{0}, I)} \Big[\big\| \mathbf{h}_{\boldsymbol{\theta}}(\alpha_t \mathbf{x}_0 + \sigma_t \mathbf{z}, t) \big\|^2 \Big].$$

- Let $\hat{\mathbf{x}}_0(\mathbf{x}_t)$ denotes the expected $\mathbf{x}_0$ given $\mathbf{x}_t$ computed by the neural network.

$$\hat{\mathbf{x}}_0(\mathbf{x}_t) = \mathbf{h}_{\boldsymbol{\theta}}(\mathbf{x}_t, t).$$

- It follows that we may estimate the score of p_t as follows:

$$\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) \approx \frac{\alpha_t \hat{\mathbf{x}}_0(\mathbf{x}_t) - \mathbf{x}_t}{\sigma_t^2}. \tag{7}$$

2.3 Denoising Diffusion Implicit Model (DDIM)

- Given a stochastic process $\{\mathbf{X}_t : 0 \le t \le T\}$ where Equation (4) holds, Song et al. gives an alternative process where $\mathbf{X}_t$ can be generated [SME22].

  1. Sample $\mathbf{X}_0 \sim p_0$.
  2. Sample $\mathbf{X}_T \sim p(\mathbf{X}_t|\mathbf{X}_0) = \mathcal{N}(\mathbf{X}_t; \alpha_T \mathbf{X}_0, \sigma_T^2 I)$.
  3. Given that some $t' > t$ has been sampled, sample $\mathbf{X}_t$ accoding to $p(\mathbf{X}_t|\mathbf{X}_0, \mathbf{X}_{t'})$, which is given by

$$p(\mathbf{X}_t|\mathbf{X}_0, \mathbf{X}_{t'}) = \mathcal{N}\bigg(\mathbf{X}_t; \alpha_t \mathbf{X}_0 + \sqrt{\sigma_t^2 - \varsigma^2} \frac{\mathbf{X}_{t'} - \alpha_{t'} \mathbf{X}_0}{\sigma_{t'}}, \varsigma^2 I\bigg)$$

  This process works for any value of $\varsigma \ge 0$.

- In particular, if $\varsigma = 0$, then we simply set

$$\mathbf{X}_t \leftarrow \alpha_t \mathbf{X}_0 + \sigma_t \frac{\mathbf{X}_{t'} - \alpha_{t'} \mathbf{X}_0}{\sigma_{t'}}. \tag{8}$$

  In fact, $\mathbf{X}_t$ for all $0 < t < T$ is determistic given we have sampled $\mathbf{X}_0$ and $\mathbf{X}_T$.

- However, the process we just described cannot be used to sample from p_0 because it requires being able to sample from p_0 in the first place.

- Song et al. proposes a practical sampling process by changing Equation (8) to use $E[\mathbf{X}_0|\mathbf{X}_{t'}]$ in place of $\mathbf{X}_0$.

$$\mathbf{X}_t \leftarrow E[\mathbf{X}_0|\mathbf{X}_{t'}] + \sigma_t \frac{\mathbf{X}_{t'} - \alpha_{t'} E[\mathbf{X}_0|\mathbf{X}_{t'}]}{\sigma_{t'}} \tag{9}$$

  Now, there is no dependency on $\mathbf{X}_0$ any more. In fact, we can use a diffusion model $\mathbf{h}_{\boldsymbol{\theta}}$ to estimate $E[\mathbf{X}_0|\mathbf{X}_{t'}]$.

- The RHS of Equation (9) is an important operation, so let's give it a name. Let

$$\mathrm{UncondDDIM}(\mathbf{x}_{t'}, \hat{\mathbf{x}}_0, t', t) = \alpha_t \hat{\mathbf{x}}_0 + \sigma_t \frac{\mathbf{x}_{t'} - \alpha_{t'}\hat{\mathbf{x}}_0}{\sigma_{t'}} = \bigg(\alpha_t - \frac{\alpha_{t'}}{\sigma_{t'}}\bigg)\hat{\mathbf{x}}_0 + \frac{\sigma_t}{\sigma_{t'}}\mathbf{x}_{t'}$$

- The **DDIM sampling** algorithm uses the UncondDDIM operation to sample from p_0. It require a series of times $0 = t_0 < t_1 < t_2 < \dots < t_K = T$, and it goes as follows.

```
Sample x_K ~ N(0, σ_T^2 I).
for k ← K downto 1 do
    x̂_0 ← h_θ(x_k, t_k)
    x_{k-1} ← UncondDDIM(x_k, x̂_0, t_k, t_{k-1})
end for
return x_0
```

2.4 Conditional Sampling

- The goal of inverse problem is to sample from $p_0(\cdot|\mathbf{y})$ assuming $\mathbf{Y} = \mathcal{A}(\mathbf{X}_0) + \sigma_{\mathbf{y}}\mathbf{Z}$ with $\mathbf{Z} \sim \mathcal{N}(\mathbf{0}, I)$.
- We can use the framework for diffusion models to do the job. In particular, we can sample from $p_0(\cdot|\mathbf{y})$ by solving a variant of Equation 2:

$$\mathrm{d}\mathbf{X}_t = \Big(\mathbf{f}(\mathbf{X}_t, t) - g^2(t)\nabla_{\mathbf{X}_t} \log p_t(\mathbf{X}_t|\mathbf{Y} = \mathbf{y})\Big)\,\mathrm{d}t + g(t)\,\mathrm{d}\overline{\mathbf{W}}_t.$$

  Equivalently, we can also solve a similar variant of the probability flow ODE (3):

$$\frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}_t, t) - \frac{g^2(t)}{2}\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t|\mathbf{Y} = \mathbf{y}). \tag{10}$$

- One can see that we require the **conditional score** $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t|\mathbf{Y} = \mathbf{y})$.
  - For brevity, we shall write it as $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t|\mathbf{y})$.
- For supervised approach, we train $\mathbf{h}_\theta(\mathbf{x}_t, t, \mathbf{y})$ so that it approximates $E[\mathbf{X}_0|\mathbf{X}_t = \mathbf{x}_t, \mathbf{Y} = \mathbf{y}]$ with the following **conditonal denosing score matching** (CDSM) loss

$$\mathcal{L}_{\mathrm{CDSM}}(\boldsymbol{\theta}) = E_{t, \mathbf{x} \sim p_{\mathbf{X}}, \mathbf{z}_{\mathbf{x}} \sim \mathcal{N}(\mathbf{0}, I_n), \mathbf{z}_{\mathbf{y}} \sim \mathcal{N}(\mathbf{0}, I_m)}\Big[\Big\|\mathbf{h}_{\boldsymbol{\theta}}(\alpha_t\mathbf{x} + \sigma_t\mathbf{z}_{\mathbf{x}}, t, \mathcal{A}(\mathbf{x}) + \sigma_{\mathbf{y}}\mathbf{z}_{\mathbf{y}}) - \mathbf{x}\Big\|^2\Big].$$

  Then, we can approximate $E[\mathbf{X}_0|\mathbf{X}_t = \mathbf{x}_t, \mathbf{Y} = \mathbf{y}]$ with

$$\hat{\mathbf{x}}(\mathbf{x}_t|\mathbf{y}) = \mathbf{h}_{\boldsymbol{\theta}}(\mathbf{x}_t, t, \mathbf{y})$$

  and $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t|\mathbf{y})$ with

$$\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t|\mathbf{y}) \approx \frac{\alpha_t\hat{\mathbf{x}}_0(\mathbf{x}_t|\mathbf{y}) - \mathbf{x}_t}{\sigma_t^2}.$$

- However, in unsupervised approaches, we only have access to a vanilla diffusion model $\mathbf{h}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)$, not a conditional diffusion model $\mathbf{h}_{\boldsymbol{\theta}}(\mathbf{x}_t, t, \mathbf{y})$. So, we have to do something else.
- Because

$$p(\mathbf{X}_t = \mathbf{x}_t|\mathbf{Y} = \mathbf{y}) = \frac{p(\mathbf{Y} = \mathbf{y}|\mathbf{X}_t = \mathbf{x}_t)p(\mathbf{X}_t = \mathbf{x_t})}{p(\mathbf{Y} = \mathbf{y})}.$$

  It follows that

$$\nabla_{\mathbf{x}_t} \log p(\mathbf{X}_t = \mathbf{x}_t|\mathbf{Y} = \mathbf{y}) = \nabla_{\mathbf{x}_t} \log p(\mathbf{Y} = \mathbf{y}|\mathbf{X}_t = \mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p(\mathbf{X}_t = \mathbf{x_t}) - \nabla_{\mathbf{x}_t} \log p(\mathbf{Y} = \mathbf{y})$$

  In other words,

$$\nabla_{\mathbf{x}_t} \log p_t ask(\mathbf{x}_t|\mathbf{y}) = \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) \tag{11}$$

- This means that we can rewrite the reverse time SDE as

$$\mathrm{d}\mathbf{X}_t = \Big(\mathbf{f}(\mathbf{X}_t, t) - g^2(t)\big(\nabla_{\mathbf{X}_t} \log p(\mathbf{y}|\mathbf{X}_t) + \nabla_{\mathbf{X}_t} \log p_t(\mathbf{X}_t)\big)\Big)\,\mathrm{d}t + g(t)\,\mathrm{d}\overline{\mathbf{W}}_t$$

  and the probability flow ODE as

$$\frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}_t, t) - \frac{g^2(t)}{2}\big(\nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)\big).$$

- We already know how to approximate $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$ with a diffusion model. So, all we have left to do is to approximate the gradient of the log-likelihood $\nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)$.

- However, the likelihood is given by an integral

$$p(\mathbf{y}|\mathbf{x}_t) \int p(\mathbf{y}|\mathbf{x}_0) p_0(\mathbf{x}_0|\mathbf{x}_t)\,\mathrm{d}\mathbf{x}_0.$$

  This is hard to compute, and it can be shown that doing so is intractible [GJP+24]. So, we have to do something else.

3 Taxonomy of Methods

- The survey paper includes about 30 methods or so.

- It attempts to categorize these according to three sets of criteria.

  - **What the method does.** This is the main criterion. There are 5 categories.
    1. **Explicit approximation for likelihood term.** Approximate $\nabla \log p(\mathbf{y}|\mathbf{x}_t)$ with a closed-form expression.
    2. **Variational inference.** Approximate the posterior $p(\mathbf{x}|\mathbf{y})$ with a simpler, tractable, parameterized distribution. The parameters of these distribution must then be optimized.
    3. **CSGM-type methods.** Optimize the noise $\mathbf{x}_T$ used to start the diffusion process.
    4. **Asymptotically exact methods.** Try to sample from the true posterior distribution $p(\mathbf{x}|\mathbf{y})$ by either using Markov chain Monte Carlo (MCMC) or sequential Monte Carlo (SMC).
    5. **Others.** This category houses methods that do not fall into the previous categories.
  - **Optimization techniques used.**
    1. **Grad.** Update $\mathbf{x}_t$ with a gradient.
    2. **Proj.** Project $\mathbf{x}_t$ or $E[\mathbf{x}_0|\mathbf{x}_t]$ to the "measurement subspace."
    3. **Samp.** Sample the next particle by defining a proposal distribution and propagate multiple chains of particles.
    4. **Opt.** Define and solve an optimization problem at every step of the sampling process. Alternatively, define a global optimization problem that emcompasses all timesteps and then solve it.

    A method can use more than one optimization techniques and so can belong to multiple categories.
  - **Type of inverse problem the method can solve.**
    * An inverse problem can be linear or non-linear based on whether the operator $\mathcal{A}$ is linear or not.
    * An inverse problem may not have noise ($\sigma_{\mathbf{y}} = 0$) or have it ($\sigma_{\mathbf{y}} > 0$).
    * An inverse problem can be blind or non-blind based on whether information on $\mathcal{A}$ is available or not. In a typically blind problems, the type of operator $\mathcal{A}$ is known, but its parameters are unknown.

4 Explicit Approximation to the Likelihood Term

- In general, the approximation has the following form

$$\nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t) \approx -\frac{\mathcal{L}_t \mathcal{M}_t}{\mathcal{G}_t}.$$

  Here,

  - $\mathcal{M}_t \in \mathbb{R}^m$ represents an error vector that measures the discrepancy between the real measurement $\mathbf{y}$ and the measurement restored from $\mathbf{x}_t$.
  - $\mathcal{L}_t \in \mathbb{R}^{n \times m}$ is a matrix the projects the error vector $\mathcal{M}_t \in \mathbb{R}^m$ back into $\mathbb{R}^n$.
  - $\mathcal{G}_t$ is a scalar scaling factor.

4.1 Score-Based Annealed Langevin Dynamics (Score ALD)

- Proposed by Jalal et al. [JAD+21].
- Works only on linear problems with noise.
  - Let $\mathcal{A}(\mathbf{x}) = A\mathbf{x}$ where $A \in \mathbb{R}^{m \times n}$.
- Makes the following approximation:

$$\nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t) \approx -\frac{A^T(\mathbf{y} - A\mathbf{x}_t)}{\sigma_{\mathbf{y}}^2 + \gamma_t^2}$$

  where γ_t is a parameter to be tuned.

- The method guides the sample in the opposite direction of the measurement error $\mathcal{M}_t = \mathbf{y} - A\mathbf{x}_t$ after being projected back with $\mathcal{L}_t = A^T$. So, this method is of category Proj.

4.2 Diffusion Posterior Sampling (DPS)

- Proposed by Chung et al. [CKM+24].
- DPS makes the following assumption.

$$\nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{X}_t = \mathbf{x}_t) \approx \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{X}_0 = E[\mathbf{X}_0|\mathbf{X}_t = \mathbf{x}_t]).$$

- Because

$$p(\mathbf{y}|\mathbf{X}_0 = E[\mathbf{X}_0|\mathbf{X}_t = \mathbf{x}_t]) = \mathcal{N}(\mathbf{y}; \mathcal{A}(E[\mathbf{X}_0|\mathbf{X}_t = \mathbf{x}_t]), \sigma_{\mathbf{y}}^2 I),$$

  we have that

$$\begin{aligned}
\nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{X}_t = \mathbf{x}_t) &= \nabla_{\mathbf{x}_t} \log \mathcal{N}(\mathbf{y}; \mathcal{A}(E[\mathbf{X}_0|\mathbf{X}_t = \mathbf{x}_t]), \sigma_{\mathbf{y}}^2 I) \\
&= \nabla_{\mathbf{x}_t} \bigg(\frac{1}{2\sigma_{\mathbf{y}}^2} \|\mathbf{y} - \mathcal{A}(E[\mathbf{X}_0|\mathbf{X}_t = \mathbf{x}_t])\| \bigg) \\
&= -\frac{1}{2\sigma_{\mathbf{y}}^2} \big(\nabla_{\mathbf{x}_t} \mathcal{A}(E[\mathbf{X}_0|\mathbf{X}_t = \mathbf{x}_t]) \big)^T \big(\mathbf{y} - \mathcal{A}(E[\mathbf{X}_0|\mathbf{X}_t = \mathbf{x}_t]) \big).
\end{aligned}$$

  So,

  - $\mathcal{G}_t = 1/(2\sigma_{\mathbf{y}}^2)$,

– $\mathcal{L}_t = \left(\nabla_{\mathbf{x}_t}\mathcal{A}(E[\mathbf{X}_0|\mathbf{X}_t = \mathbf{x}_t])\right)^T$, and
– $\mathcal{M}_t = \mathbf{y} - \mathcal{A}(E[\mathbf{X}_0|\mathbf{X}_t = \mathbf{x}_t])$.

- In practice, DPS does not use $\mathcal{G}_t = 1/(2\sigma_{\mathbf{y}}^2)$. Instead, it proposes using scaling factor

$$\mathcal{G}_t = \zeta_t = \frac{\zeta'}{\|\mathbf{y} - \mathcal{A}(E[\mathbf{X}_0|\mathbf{X}_t = \mathbf{x}_t])\|}$$

where ζ' is a constant.

References

[And82] B. D. Anderson, *Reverse-time diffusion equation models*, Stochastic Processes and their Applications **12** (1982), no. 3, 313–326.

[BM18] Y. Blau and T. Michaeli, *The perception-distortion tradeoff*, 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, June 2018, p. 6228–6237.

[CKM+24] H. Chung, J. Kim, M. T. Mccann, M. L. Klasky, and J. C. Ye, *Diffusion posterior sampling for general noisy inverse problems*, 2024, `arXiv:2209.14687 [stat.ML]`.

[DCL+24] G. Daras, H. Chung, C.-H. Lai, Y. Mitsufuji, J. C. Ye, P. Milanfar, A. G. Dimakis, and M. Delbracio, *A survey on diffusion models for inverse problems*, 2024, `arXiv:2410.00083 [cs.LG]`.

[DM24] M. Delbracio and P. Milanfar, *Inversion by direct iteration: An alternative to denoising diffusion for image restoration*, 2024, `arXiv:2303.11435 [eess.IV]`.

[GJP+24] S. Gupta, A. Jalal, A. Parulekar, E. Price, and Z. Xun, *Diffusion posterior sampling is computationally intractable*, 2024, `arXiv:2402.12727 [cs.LG]`.

[JAD+21] A. Jalal, M. Arvinte, G. Daras, E. Price, A. G. Dimakis, and J. I. Tamir, *Robust compressed sensing mri with deep generative priors*, 2021, `arXiv:2108.01368 [cs.LG]`.

[KAAL22] T. Karras, M. Aittala, T. Aila, and S. Laine, *Elucidating the design space of diffusion-based generative models*, 2022, `arXiv:2206.00364 [cs.CV]`.

[LGZ+23a] Z. Luo, F. K. Gustafsson, Z. Zhao, J. Sjölund, and T. B. Schön, *Image restoration with mean-reverting stochastic differential equations*, 2023, `arXiv:2301.11699 [cs.LG]`.

[LGZ+23b] ———, *Refusion: Enabling large-size realistic image restoration with latent-space diffusion models*, 2023, `arXiv:2304.08291 [cs.CV]`.

[LYC+21] H. Li, Y. Yang, M. Chang, H. Feng, Z. Xu, Q. Li, and Y. Chen, *Srdiff: Single image super-resolution with diffusion probabilistic models*, 2021, `arXiv:2104.14951 [cs.CV]`.

[SCC+22] C. Saharia, W. Chan, H. Chang, C. A. Lee, J. Ho, T. Salimans, D. J. Fleet, and M. Norouzi, *Palette: Image-to-image diffusion models*, 2022, `arXiv:2111.05826 [cs.CV]`.

[SHC+21] C. Saharia, J. Ho, W. Chan, T. Salimans, D. J. Fleet, and M. Norouzi, *Image super-resolution via iterative refinement*, arXiv:2104.07636 (2021).

[SME22] J. Song, C. Meng, and S. Ermon, *Denoising diffusion implicit models*, 2022, `arXiv:2010.02502 [cs.LG]`.

[SSDK+21] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, *Score-based generative modeling through stochastic differential equations*, 2021, `arXiv:2011.13456 [cs.LG]`.

[WDT+21] J. Whang, M. Delbracio, H. Talebi, C. Saharia, A. G. Dimakis, and P. Milanfar, *Deblurring via stochastic refinement*, 2021, `arXiv:2112.02475 [cs.CV]`.

[ZRA23] L. Zhang, A. Rao, and M. Agrawala, *Adding conditional control to text-to-image diffusion models*, 2023, `arXiv:2302.05543 [cs.CV]`.