Diffusion for inverse problem

1. Generative models

Generative models are trying to find the joint distribution $q(x)$, where x can be images, texts or audios. In most cases, we cannot have access to the real joint distribution $q(x)$, but we do have training datasets, $X = x_1, ..., x_N$ and we assume every datapoint in our dataset is sampled from real distribution iid. $x_i \sim q(x)$. The goal of generative models is to learn a $p_\theta(x)$ that approximate $q(x)$, where $p_\theta(X)$ will have high probability, maximize datapoint likelihood. Common approaches are like Bayesian network(VAE), auto-regressive models, and flow models. However, there are methods that does not rely on joint prob estimation, like GANS. Those methods will rely on adversarial training, and will be a nightmare during training, and will have mode collapse during sampling.

1.1 Why do we need generative models?

So why do we need generative models in the first place? Remember Google Deep Dream? Suppose if we have a trained classifier, $p_\theta(y|x)$, can we use SGD to find the perfect x for any given y? $\operatorname{argmax}_x p_\theta(y|x)$. The short answers is no, since $p(x|y) = \frac{p(y|x)p(x)}{p(y)}$ and $p(x)$ is the term(prior) that make our generative content looks real.

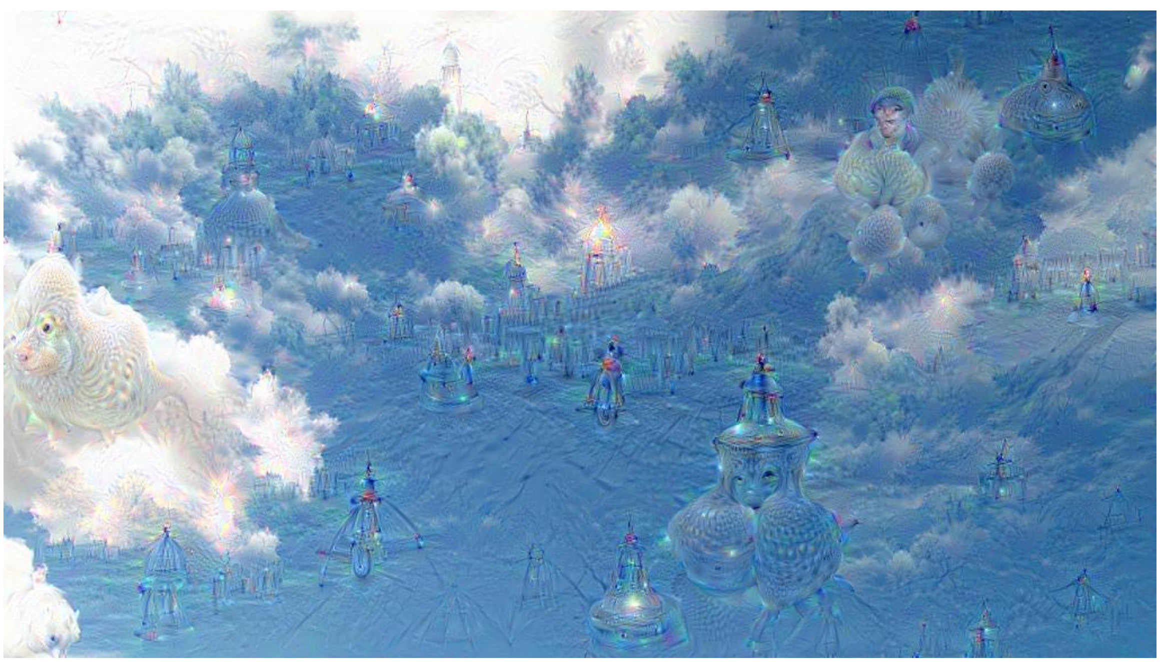

Figure 1: An image generated by deep dream.

2. Diffusion models

In this section, we will introduce 2 basic diffusion models **DDPM** (Ho et al., 2020) and **score based diffusion** (Song et al., 2021). Think those two are a discrete time-step(DDPM) and a continuous processes(Score based). We will build connections later.

2.1 DDPM

First, lets define some notations,

- x_0 is real data from dataset.
- $q(x)$ is the real distribution, which we don't know.
- $\{\beta_t \in (0,1)\}_{t=1}^T$ are some constant coefficients.
- $\alpha_t = 1 - \beta t$.
- $\overline{\alpha}_t = \Pi_{i=1}^t \alpha_i$
- $\epsilon_* \sim \mathcal{N}(0, \mathbf{I})$ are some noises.

some assumed properties,

- $q(x_{1:T}|x_0) = \Pi_{t=1}^T q(x_t|x_{t-1})$, (**Markov property**).
- $q(x_t|x_{t-1}) = \mathcal{N}(x_t|\sqrt{1-\beta_t}x_{t-1}, \beta_t \mathbf{I})$, **forward process**(adding noise) defined by diffusion model.

With everything above, we can derived some important and convenient properties. First, by chaining up $q(x_t|x_{t-1})$ from 1 to t, we have the following:

$$x_t = \sqrt{\overline{\alpha}_t}x_0 + \sqrt{1-\overline{\alpha}_t}\epsilon. \tag{1}$$

And our goal is to learn or approximate the **reverse process**, $q(x_{t-1}|x_t)$. This is the hard part and we don't have closed form of it. However, we do have the closed form of conditional reverse process.

$$q(x_{t-1}|x_t, x_0) = q(x_t|x_{t-1}, x_0)\frac{q(x_{t-1}|x_0)}{q(x_t|x_0)} \tag{2}$$

$$= \mathcal{N}(x_{t-1}|\tilde{\mu}(x_t, x_0), \tilde{\beta}_t \mathbf{I}). \tag{3}$$

where $\tilde{\mu}(x_t, x_0) = \frac{\sqrt{\alpha_t}(1-\overline{\alpha}_{t-1})}{1-\overline{\alpha}_t}x_t + \frac{\sqrt{\overline{\alpha}_{t-1}}\beta_t}{1-\overline{\alpha}_t}x_0 = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\epsilon_t\right)$, and $\tilde{\beta}_t = \frac{1-\overline{\alpha}_{t-1}}{1-\overline{\alpha}_t}\beta_t$. Why is this not good enough? Because if we chain up the conditional reverse process, we will always return back exactly or close to x_0. Also in sampling, we don't have access to x_0.

So the only step left is to approximate $q(x_{t-1}|x_t)$ with $p_\theta(x_{t-1}|x_t)$. Lets define some properties of $p_\theta(x_{t-1}|x_t)$ as follows,

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}|\mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \tag{4}$$

$$p_\theta(x_{0:T}) = p_\theta(x_T)\Pi_{t=1}^T p_\theta(x_{t-1}|x_t), \ \textbf{Markov property.} \tag{5}$$

Loss. We can derive loss objective by minimizing NLL, $-\log p_\theta(x_0)$.

$$-\log p_\theta(x_0) \leq -\log p_\theta(x_0) + \mathbf{D}_{KL}\left(q(x_{1:T}|x_0)||p_\theta(x_{1:T}|x_0)\right) \tag{6}$$
$$= \mathbf{E}_{x_{0:T}\sim q(\cdot)}\left[\log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})}\right] \tag{7}$$

Hence minimizing the upper bound will minimize NLL. Why not use the KL in reverse fashion, $\mathbf{D}_{KL}\left(p_\theta(x_{1:T}|x_0)||q(x_{1:T}|x_0)\right)$? The easy answer is sampling from p_θ does not make sense, and our dataset is sampled from q. Also, by doing this, we will have a lower bound, and optimizing lower bound will not minimize NLL.
Minimize (6) is not straight forward, unless we split the terms as follow:

$$\mathbf{E}_{x_{0:T}\sim q(\cdot)}\left[\log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})}\right] = \mathbf{D}_{KL}\left(q(x_T)||p_\theta(x_T)\right) + \sum_{t=2}^{T}\mathbf{D}_{KL}\left(q(x_{t-1}|x_t,x_0)||p_\theta(x_{t-1}|x_t)\right)) - \log p_\theta(x_0|x_1) \tag{8}$$

The first term can be ignored, since x_T is pure Gaussian noise and that term is a constant. The rest terms can be optimized with L2 loss of $\tilde{\mu}(x_t, x_0)$ and $\mu_\theta(x_t, y)$, given $q(x_{t-1}|x_t, x_0)$ and $p_\theta(x_{t-1}|x_t))$ are uni-modal Gaussian. Training and sampling is quite clear as shown in:

Algorithm 1 Training

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

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0},\mathbf{I})$
2: **for** $t = T,\dots,1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0},\mathbf{I})$ if $t>1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t,t)\right) + \sigma_t\mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

Figure 2: The training and sampling algorithms in DDPM (Image source: (Ho et al., 2020))

2.2 Score based diffusion

As discussed earlier, generative model is dealing with joint prob $p_\theta(x) \approx q(x)$. $p(x)$ is essentially a pdf, so it must have form of

$$p_\theta(x) = \frac{\exp(-f_\theta(x))}{\mathbf{Z}_\theta} \tag{9}$$

and the normalization term $\mathbf{Z}_\theta$ is hard to model. But we can avoid this by taking the derivative wrt x.

$$s_\theta(x) = \nabla_x \log p_\theta(x) = -\nabla_x f_\theta(x) - \nabla_x log\mathbf{Z}_\theta = -\nabla_x f_\theta(x) \tag{10}$$

So our goal is to approximate $\nabla_x \log q(x)$ with $s_\theta(x)$. We can use Fisher divergence,

$$\mathbf{D}_F = \mathbf{E}_{x\sim q(x)}\left[\|\nabla_x \log q(x) - s_\theta(x)\|_2^2\right] \tag{11}$$

Again, we can only sample from q, and do not have closed form of it, hence we need to use **score matching**, which can bypass the need of ground truth data score.
According to Score matching (Hyvärinen, 2005), minimizing Fisher divergence, is the same as minimizing the following objective,

$$\mathbf{E}_{x\sim q(x)}\left[\frac{1}{2}\|s_\theta(x)\|_2^2 + trace(\nabla_x s_\theta(x))\right] \tag{12}$$

The Jacobian requires backprop and is not scalable on high dim. A scalable approach is to use random projection to 1D (Song et al., 2019). Then the objective(**Sliced SM**) is

$$\mathbf{E}_{v\sim p(v)}\mathbf{E}_{x\sim q(x)}\left[v^\intercal \nabla_x s_\theta(x) v + \frac{1}{2}(v^\intercal s_\theta)^2\right] \tag{13}$$

Another way is to use Denoising SM (Vincent, 2011), where we define a perturbation kernel $q_\sigma(\tilde{x}|x)$, and then the objective is

$$\frac{1}{2}\mathbf{E}_{x\sim q(x)}\mathbf{E}_{\tilde{x}\sim q_\sigma(\cdot|x)}\left[\|\nabla_{\tilde{x}}\log q_\sigma(\tilde{x}|x) - s_\theta(\tilde{x})\|_2^2\right] \tag{14}$$

Now the score function s_θ is approximating the score of noisy version of data distribution. And it cannot estimate the score of noise-free distribution. So in practice, we need to choose a small noise, but this will make the variance of this objective explode.
Suppose we have a perfect score function, the generating samples is easy,

$$\tilde{x}_{t+1} \leftarrow \tilde{x}_t + \frac{\epsilon}{2}s_\theta(\tilde{x}_t) \tag{15}$$

$$\tilde{x}_{t+1} \leftarrow \tilde{x}_t + \frac{\epsilon}{2}s_\theta(\tilde{x}_t) + \sqrt{\epsilon}z_t \tag{16}$$

(15) is standard SGD, which will lead samples to collide together. (16) is SGLD, which will generate correct but different samples. In reality, we will not have correct score function in low density regions, and Langevin dynamics will have trouble exploring low density(inaccurate) regions. Hence, multiple noise level perturbation is needed. Let's jump to the objective function first,

$$\frac{1}{N}\lambda(\sigma_i)\mathbf{E}_{x\sim q_{\sigma_i}(x)}\left[\|\nabla_x \log q_{\sigma_i}(x) - s_\theta(x,\sigma_i)\|_2^2\right] \tag{17}$$

Optimizing (17) is similar as in (14). And once we have $s_\theta(x,\sigma_i)$ trained, sampling with annealed Langevin dynamics will be less troublesome.
Conditional generation. Once we have score of data distribution $\nabla_x q(x)$, then generation from the **inverse distribution**, $p(x|y)$, would be simple. From Bayes rule, $p(x|y) = \frac{p(x)p(y|x)}{p(y)}$, and hence:

$$\begin{aligned}
\nabla_x p(x|y) &= \nabla_x \log p(x) + \nabla_x \log(y|x) - \nabla_x \log p(y) && (18)\\
&= \nabla_x \log p(x) + \nabla_x \log p(y|x) && (19)\\
&\approx s_\theta(x) + \nabla_x \log p(y|x) && (20)
\end{aligned}$$

In order to generalize score based diffusion model to infinite noise level, (Song et al., 2021) defined it based on SDE. $\{X_t\}_{t\in[0,T]}$ is a stochastic process, and X_t is a random variable. For each RV X_t, there is a PDF, $q_t(X)$. And $q_0(x) = q(x)$ which is real data distribution, and $q_T(x) = \pi(x)$ is close to Gaussian distribution. In forward process, we can define an SDE of following form,

$$dx = f(x,t)dt + g(t)dw \tag{21}$$

and the reverse process is

$$dx = \left[f(x,t) + g^2(t)\nabla_x \log q_t(x)\right] dt + g(t)d\overline{w}. \tag{22}$$

During training, they used DSM objective,

$$\min_\theta \mathbb{E}_{t\sim\mathcal{U}(0,T)}[\lambda(t)\mathbb{E}_{\mathbf{x}(0)\sim q_0(\mathbf{x})}\mathbf{E}_{\mathbf{x}(t)\sim q_{0t}(\mathbf{x}(t)|\mathbf{x}(0))}[\|s_\theta(\mathbf{x}(t),t) - \nabla_{\mathbf{x}(t)} \log q_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0))\|_2^2]], \tag{23}$$

and $\log q_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0))$ depend on choice of f and g.
For sampling, Euler-Maruyama is one easy approach, where we can replace dt with Δt and $d\overline{w}$ with $z \sim \mathcal{N}(0, g^2(t)\Delta t\mathbf{I})$.

2.3 Connection between DDPM and Score based diffusion

In (Song et al., 2021), VP-SDE is exactly the same as DDPM, just with continuous noise level. For simple explanation, suppose $x \sim \mathcal{N}(\mu, \sigma^2\mathbf{I})$, then $\nabla_x \log p(x) = -\frac{\epsilon}{\sigma}$, where $\epsilon \sim \mathcal{N}(0,\mathbf{I})$. Recall that in DDPM, $q(x_t|x_0) = \mathcal{N}(\sqrt{\overline{\alpha}_t}x_0, (1-\overline{\alpha}_t)\mathbf{I})$, then we can show that,

$$\begin{aligned} s_\theta(x_t,t) &\approx \nabla_{x_t} \log q(x_t) &(24)\\ &= \mathbf{E}_{q(x_0)}\left[\nabla_{x_t} q(x_t|x_0)\right] &(25)\\ &= \mathbf{E}_{q(x_0)}\left[-\frac{\epsilon_\theta(x_t,t)}{\sqrt{1-\overline{\alpha}_t}}\right] &(26)\\ &= -\frac{\epsilon_\theta(x_t,t)}{\sqrt{1-\overline{\alpha}_t}} &(27)\end{aligned}$$

Estimating score, ϵ, or x_0 is the same thing!

2.4 DDIM

At glance on DDIM (Song et al., 2022), I believed it is just a faster way for sampling. To make it simple, lets derive $q(x_{t-1}|x_t,x_0)$ from eqs(1).

$$\begin{aligned} q(x_{t-1}|x_t,x_0) &= \sqrt{\overline{\alpha}_{t-1}}x_0 + \sqrt{1-\overline{\alpha}_{t-1}}\epsilon &(28)\\ &\approx \sqrt{\overline{\alpha}_{t-1}}\hat{x}_{0|t} + \sqrt{1-\overline{\alpha}_{t-1}}\epsilon &(29)\\ &\approx \sqrt{\overline{\alpha}_{t-1}}\hat{x}_{0|t} + \sqrt{1-\overline{\alpha}_{t-1}}\epsilon_\theta(x_t,t) &(30)\\ &\approx \sqrt{\overline{\alpha}_{t-1}}\hat{x}_{0|t} + \sqrt{1-\overline{\alpha}_{t-1}-\sigma_t^2}\epsilon_\theta + \sigma_t\epsilon &(31)\end{aligned}$$

However, the main question in DDIM is one whether diffusion model has to be Markovian? Put in other words, can we derive $q(x_{t-1}|x_t,x_0)$, only based on $q(x_t|x_0)$ and $q(x_{t-1}|x_0)$, without $q(x_t|x_{t-1})$?

$$\begin{cases} x_{t-1} &= m_t x_t + n_t x_0 + \sigma_t\epsilon_1 \\ x_t &= \sqrt{\overline{\alpha}_t}x_0 + \sqrt{1-\overline{\alpha}_t}\epsilon_2 \\ x_{t-1} &= \sqrt{\overline{\alpha}_{t-1}}x_0 + \sqrt{1-\overline{\alpha}_{t-1}}\epsilon_3 \end{cases} \tag{32}$$

If we solve (31) and represent m_t and n_t using σ_t and plug it back, we can get eqs(30). NOTE, if $\sigma_t = 0$, then generative process is deterministic, final sample only depends on x_T, so we can treat it as a latent value.

3. Diffusion models for inverse problem

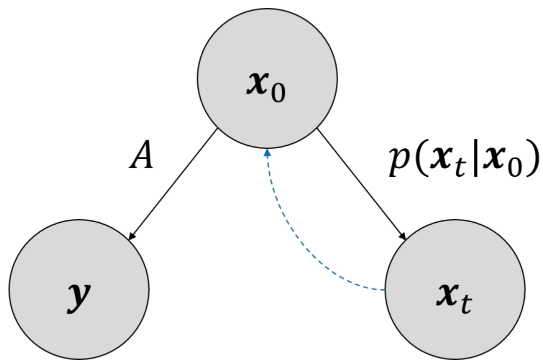

Figure 3: Bayes net of inverse problem (Image source: (Chung et al., 2023))

Inverse problem in diffusion models can be divided into two categories, (known) inverse problem[(Kawar et al., 2022), (Chung et al., 2023)] and blind inverse problem[(Chung et al., 2022), (Laroche et al., 2023)]. The general problem set up is as follows,

$$y = \mathbf{A}(x) + \eta \tag{33}$$

For an inverse problem in CV community, eventually they want to sample from posterior distribution $p(x|y)$. This posterior can be viewed as impainting, debluring problem.

4. Known inverse problem

This problem is quite straight forward, since we know $\mathbf{A}$. According to eqs 19, the only unknown is $\nabla_{x_t} \log p(y|x_t)$. As shown in (Chung et al., 2023), we can use Tweedie's formula, and

$$p(y|x_t) \approx p(y|\hat{x}_0) \tag{34}$$

where $\hat{x}_0 \approx \frac{1}{\sqrt{\overline{\alpha}(t)}} [x_t + (1 - \overline{\alpha}(t))s_\theta(x_t, t)]$ is the approximation of x_0 at time step t. Why we don't have $p(y|x_t)$? Because the generation process of y as shown in eqs 32, where x is sampled from real distribution $x \sim p(x)$. And if η is Gaussian, we have

$$\nabla_{x_t} \log p(y|x_t) \approx -\frac{1}{\sigma^2} \nabla_{x_t} \|y - \mathbf{A}(\hat{x}_0(x_t))\|_2^2 \tag{35}$$

And the final algorithm is 4, Up to step 6, there is no difference than normal diffusion model, step 7 can be interpreted as a diffusion direction towards y. In other words, step 6 will lead final sample $\hat{x}_0$ looks real, and step 7 will lead to $y \approx \mathbf{A}(\hat{x}_0) + \eta$.

Algorithm 1 DPS - Gaussian

Require: N, $\boldsymbol{y}$, $\{\zeta_i\}_{i=1}^N$, $\{\tilde{\sigma}_i\}_{i=1}^N$
$\boldsymbol{x}_N \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
for $i = N-1$ **to** 0 **do**
  $\hat{\boldsymbol{s}} \leftarrow \boldsymbol{s}_\theta(\boldsymbol{x}_i, i)$
  $\hat{\boldsymbol{x}}_0 \leftarrow \frac{1}{\sqrt{\bar{\alpha}_i}}(\boldsymbol{x}_i + (1-\bar{\alpha}_i)\hat{\boldsymbol{s}})$
  $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
  $\boldsymbol{x}'_{i-1} \leftarrow \frac{\sqrt{\alpha_i}(1-\bar{\alpha}_{i-1})}{1-\bar{\alpha}_i}\boldsymbol{x}_i + \frac{\sqrt{\bar{\alpha}_{i-1}}\beta_i}{1-\bar{\alpha}_i}\hat{\boldsymbol{x}}_0 + \tilde{\sigma}_i \boldsymbol{z}$
  $\boldsymbol{x}_{i-1} \leftarrow \boldsymbol{x}'_{i-1} - \zeta_i \nabla_{\boldsymbol{x}_i} \|\boldsymbol{y} - \mathcal{A}(\hat{\boldsymbol{x}}_0)\|_2^2$
end for
return $\hat{\mathbf{x}}_0$

Algorithm 2 DPS - Poisson

Require: N, $\boldsymbol{y}$, $\{\zeta_i\}_{i=1}^N$, $\{\tilde{\sigma}_i\}_{i=1}^N$
$\boldsymbol{x}_N \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
for $i = N-1$ **to** 0 **do**
  $\hat{\boldsymbol{s}} \leftarrow \boldsymbol{s}_\theta(\boldsymbol{x}_i, i)$
  $\hat{\boldsymbol{x}}_0 \leftarrow \frac{1}{\sqrt{\bar{\alpha}_i}}(\boldsymbol{x}_i + (1-\bar{\alpha}_i)\hat{\boldsymbol{s}})$
  $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
  $\boldsymbol{x}'_{i-1} \leftarrow \frac{\sqrt{\alpha_i}(1-\bar{\alpha}_{i-1})}{1-\bar{\alpha}_i}\boldsymbol{x}_i + \frac{\sqrt{\bar{\alpha}_{i-1}}\beta_i}{1-\bar{\alpha}_i}\hat{\boldsymbol{x}}_0 + \tilde{\sigma}_i \boldsymbol{z}$
  $\boldsymbol{x}_{i-1} \leftarrow \boldsymbol{x}'_{i-1} - \zeta_i \nabla_{\boldsymbol{x}_i} \|\boldsymbol{y} - \mathcal{A}(\hat{\boldsymbol{x}}_0)\|_{\boldsymbol{\Lambda}}^2$
end for
return $\hat{\mathbf{x}}_0$

Figure 4: DPS ALGO (Image source: (Chung et al., 2023))

It is more straight forward to derive eqs 33 with importance sampling.

$$
\begin{aligned}
p(y|x_t) &= \int p(y|x_0, x_t)p(x_0|x_t)dx_0 && (36)\\
&= \int p(y|x_0)p(x_0|x_t)dx_0 && (37)\\
&= \int p(y|x_0)p(x_0|x_t)\frac{q(x_0|x_t)}{q(x_0|x_t)}dx_0 && (38)\\
&= \mathbf{E}_{x_0\sim q(x_0|x_t)}\left[\frac{p(y|x_0)p(x_0|x_t)}{q(x_0|x_t)}\right] && (39)\\
&\approx \mathbf{E}_{\hat{x}_0\sim q(\hat{x}_0|x_t)}\left[p(y|\hat{x}_0)\right] && (40)\\
& && (41)
\end{aligned}
$$

5. Blind inverse problem

In blind inverse problem, we don't know the exact value of A, but in (Chung et al., 2022), they assume the class of A is known(Gaussian deblurring). For blind deblurring objective, it is

$$
y = k * x + n \tag{42}
$$

Here k is a blur kernel, and parameterized by ψ, where $y = H_\psi(x) + n$. A classic way to solve this is

$$
\min_{x,\psi} \frac{1}{2}\|H_\psi(x) - y\|^2 + \mathbf{R}_\psi(\psi) + \mathbf{R}_x(x) \tag{43}
$$

$\mathbf{R}(\cdot) = -\log(p(\cdot))$ is the regularization function or the NLL prior. In (Chung et al., 2022), they argue that eqs (36) is sub-optimal due to 1. $\mathbf{R}(\cdot)$ do not fully represent true prior. 2. Optimization process is unstable. In (Chung et al., 2022), the goal is $p(x_0, k_0|y)$ and it is proportional to

$$
p(x_0, k_0|y) \propto p(y|x_0, k_0)p(x_0)p(k_0) \tag{44}
$$

Eqs(43) only holds when x_0, k_0 are independent. They modeled $p(x_0)$ and $p(k_0)$ with two separate diffusion process. In order to sample from $p(x_0, k_0|y)$, we need to have

$$\nabla_{x_t} \log p(x_t, k_t|y) = \nabla_{x_t} \log p(y|x_t, k_t) + \nabla_{x_t} \log p(x_t) \tag{45}$$

$$\nabla_{k_t} \log p(x_t, k_t|y) = \nabla_{k_t} \log p(y|x_t, k_t) + \nabla_{k_t} \log p(k_t) \tag{46}$$

The first term in RHS is not tractable, and need to be approximated by

$$\nabla_{x_t} \log p(y|x_t, k_t) \approx \nabla_{x_t} \log p(y|\hat{x}_0(x_t), \hat{k}_0(k_t)) \tag{47}$$

$$\nabla_{k_t} \log p(y|x_t, k_t) \approx \nabla_{k_t} \log p(y|\hat{x}_0(x_t), \hat{k}_0(k_t)) \tag{48}$$

The BlindDPS algorithm is as follow, Here is the ablation test between uniform prior and BlindDPS.

Algorithm 1 BlindDPS — Blind Deblurring

Require: $N, \boldsymbol{y}, \alpha, \{\tilde{\sigma}_i\}_{i=1}^N, \lambda, R_{\boldsymbol{k}}(\cdot)$

$\boldsymbol{x}_N, \boldsymbol{k}_N \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$
for $i = N-1$ **to** 0 **do**
  $\hat{\boldsymbol{s}}^i \leftarrow \boldsymbol{s}_{\theta^*}^i(\boldsymbol{x}_i, i)$
  $\hat{\boldsymbol{s}}^k \leftarrow \boldsymbol{s}_{\theta^*}^k(\boldsymbol{k}_i, i)$
  $\hat{\boldsymbol{x}}_0 \leftarrow \frac{1}{\sqrt{\bar{\alpha}_i}}(\boldsymbol{x}_i + \sqrt{1-\bar{\alpha}_i}\hat{\boldsymbol{s}}^i)$
  $\hat{\boldsymbol{k}}_0 \leftarrow \frac{1}{\sqrt{\bar{\alpha}_i}}(\boldsymbol{k}_i + \sqrt{1-\bar{\alpha}_i}\hat{\boldsymbol{s}}^k)$
  $\hat{\boldsymbol{k}}_0 \leftarrow \mathcal{P}_C(\hat{\boldsymbol{k}}_0)$
  $\boldsymbol{z}_i, \boldsymbol{z}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$
  $\boldsymbol{x}'_{i-1} \leftarrow \frac{\sqrt{\alpha_i}(1-\bar{\alpha}_{i-1})}{1-\bar{\alpha}_i}\boldsymbol{x}_i + \frac{\sqrt{\bar{\alpha}_{i-1}}\beta_i}{1-\bar{\alpha}_i}\hat{\boldsymbol{x}}_0 + \tilde{\sigma}_i \boldsymbol{z}_i$
  $\boldsymbol{k}'_{i-1} \leftarrow \frac{\sqrt{\alpha_i}(1-\bar{\alpha}_{i-1})}{1-\bar{\alpha}_i}\boldsymbol{k}_i + \frac{\sqrt{\bar{\alpha}_{i-1}}\beta_i}{1-\bar{\alpha}_i}\hat{\boldsymbol{k}}_0 + \tilde{\sigma}_i \boldsymbol{z}_k$
  $\boldsymbol{x}_{i-1} \leftarrow \boldsymbol{x}'_{i-1} - \alpha \nabla_{\boldsymbol{x}_i} \|\boldsymbol{y} - \hat{\boldsymbol{k}}_0 * \hat{\boldsymbol{x}}_0\|_2$
  $\mathcal{L}_{\boldsymbol{k}} \leftarrow \|\boldsymbol{y} - \hat{\boldsymbol{k}}_0 * \hat{\boldsymbol{x}}_0\|_2 + \lambda R_{\boldsymbol{k}}(\hat{\boldsymbol{k}}_0)$
  $\boldsymbol{k}_{i-1} \leftarrow \boldsymbol{k}'_{i-1} - \alpha \nabla_{\boldsymbol{k}_i} \mathcal{L}_{\boldsymbol{k}}$
end for
return $\boldsymbol{x}_0, \boldsymbol{k}_0$

Figure 5: BlindDPS ALGO (Image source: (Chung et al., 2022))

6. Functional diffusion

All previous work is based on estimating $p(x)$, where x is an image. However, those cannot directly applied to solving PDE or inverse problem in PDE. Since our objective is $p(u(x))$, not $p(u)$. The only paper on this is (Zhang and Wonka, 2023).

Suppose we have a training dataset $\mathbf{D}$ contains collection of functions $f_0 : \mathbf{X} \to \mathbf{Y}$. And $\mathbf{F}$ is a function set, each element is $g : \mathbf{X} \to \mathbf{Y}$. Then we can define a noised version of f as

$$f_t(x) = \alpha_t f_0(x) + \sigma_t g(x) \tag{49}$$

The goal is to train a denoiser which approximate

$$D_\theta[f_t, t](x) \approx f_0(x) \tag{50}$$

Since neural network input cannot be functions, we need to discretize f_t. Then the loss objective is

$$w(t) \sum_{i \in \mathbf{Q}} \left| \mathbf{D}_\theta \left[\{x_j, f_t(x_j)\}_{j \in \mathbf{C}}, t, x_i \right] - f_0(x_i) \right|^2 \tag{51}$$

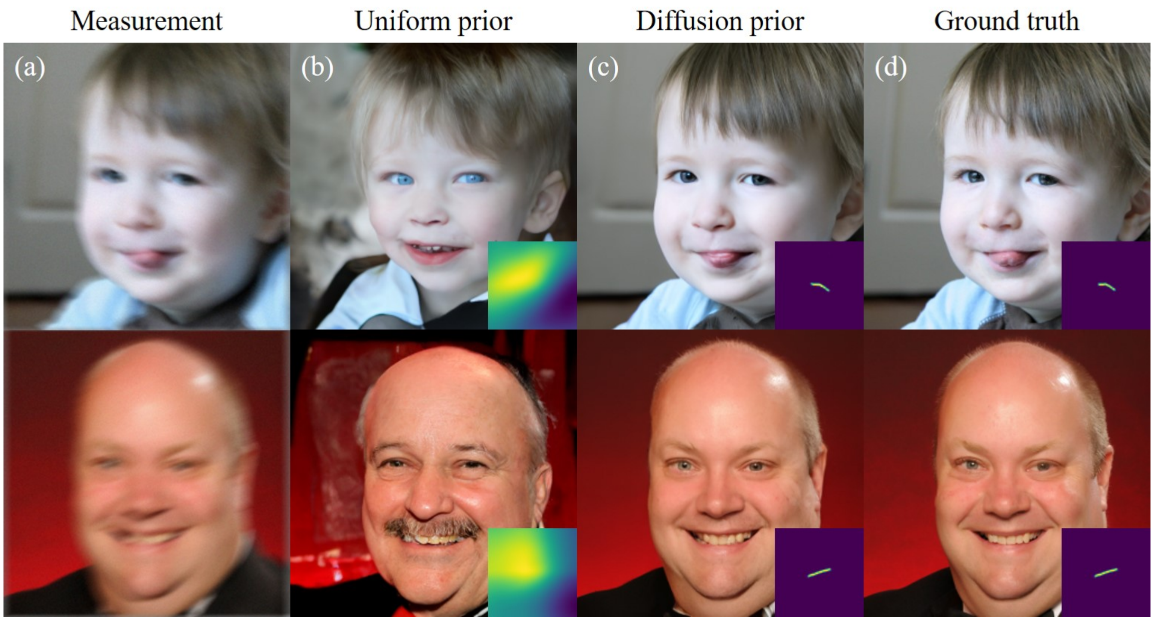

Figure 6: BlindDPS Vs Uniform prior (Image source: (Chung et al., 2022))

The sampling procedure(s<t) is

$$f_s = \alpha_s \mathbf{D}_\theta \left[f_t, t\right] + \sigma_s \left(\frac{f_t - \alpha_t \mathbf{D}_\theta[f_t, t]}{\sigma_t} \right) \tag{52}$$

$$f_s(x) = \frac{\sigma_s}{\sigma_t} f_t(x) + \left(\alpha_s - \sigma_s \frac{\alpha_t}{\sigma_t} \right) \mathbf{D}_\theta[\{x_i, f_t(x_i)\}_{i\in\mathbf{C}}, t, x] \tag{53}$$

NOTES: In inverse PDE case, $\mathbf{C}$ and $\mathbf{Q}$ are sets of collocation points. Our samples(observations) and boundary condition are conditioned in D_θ. A detailed version can be found in 7. Normally, we need to sample $g(x)$ from a GP, but that is time consuming during training. In (Zhang and Wonka, 2023), they sample Gaussian noise on a grid in $\mathbf{X}$ and interpolate other values with values on the grid. Their algorithm is shown in 8,

Some questions and uncertain staff.

1. In sampling algorithm, they let $f_t(x) = g(x)$, but in reality $f_t(x) = \alpha_t f_0(x) + \sigma_t g(x)$ and according to training algorithm, α_t will never reach 0.
2. Should observation points(condition points) be the same during sampling as in training?

7. Inverse problem in PDE

We can combine ideas from BlindDPS and functional diffusion for our PDE inverse problem. Take Darcy flow as example,

$$\begin{cases} -\nabla \cdot (a(x)\nabla u(x)) = f(x) \\ u(x) = q(x), x \in d\Omega \\ \{x_i, u(x_i)\}_{i=i}^{N} \end{cases} \tag{54}$$

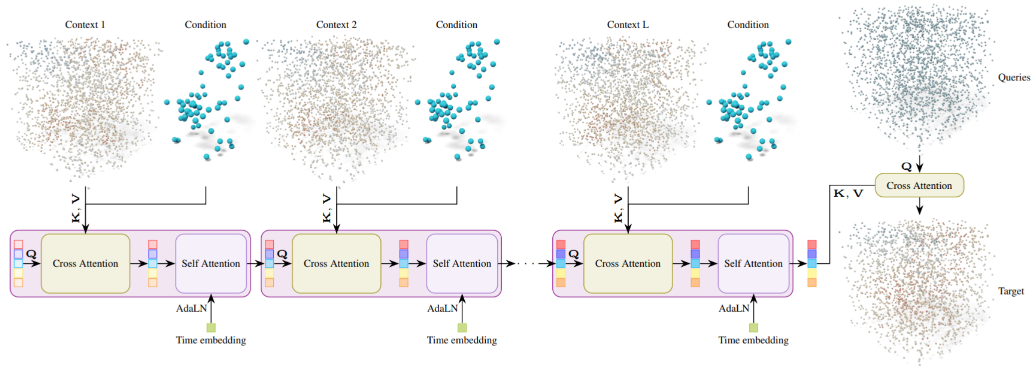

Figure 5. **The network design of the SDF diffusion model.** The context set is split into L smaller ones. They (and optionally conditions such as sparse surface point clouds) are fed into different stages of the network by using cross-attention. The time embedding is injected into the network in every self-attention layer by adaptive layer normalization. After L stages, we obtain the representation vector sets and they will be used to predict values of arbitrary queries. For SDFs, we optimize simple minimum squared errors.

Figure 7: Network design of functional diffusion. (Image source: (Zhang and Wonka, 2023))

Algorithm 1 Training

repeat
  $g \in \mathcal{F}$ ▷ noise function
  $f_0 \in \mathcal{D}$ ▷ training function
  $t \sim \mathcal{T}$ ▷ noise level
  $\alpha_t = 1/\sqrt{t^2+1}, \sigma_t = t/\sqrt{t^2+1}$ ▷ SNR
  Sample $\mathcal{C}$ ▷ context
  Evaluate $\{g(\mathbf{x}_i)\}_{i\in\mathcal{C}}$ and $\{f_0(\mathbf{x}_i)\}_{i\in\mathcal{C}}$
  Calculate the context $\{f_t(\mathbf{x}_i)\}_{i\in\mathcal{C}}$ with Eq. (3)
  Sample $\mathcal{Q}$ ▷ query
  Optimize Eq. (9) ▷ denoise
until convergence

Algorithm 2 Sampling

Ensure: Sample $\mathcal{C}$ and $g \in \mathcal{F}$
Let $f_t = g$
Evaluate $\{\mathbf{x}_i, f_t(\mathbf{x}_i)\}_{i\in\mathcal{C}}$
for $k \in \{N, N-1, \ldots, 2, 1\}$ **do**
  $t_k = T(k), t_{k-1} = T(k-1)$
  $\alpha_t = 1/\sqrt{t_k^2+1}, \alpha_s = 1/\sqrt{t_{k-1}^2+1}$
  $\sigma_t = t_k/\sqrt{t_k^2+1}, \sigma_s = t_{k-1}/\sqrt{t_{k-1}^2+1}$
  Predict $\{f_s(\mathbf{x}_i)\}_{i\in\mathcal{C}}$ with Eq. (11)
  Let $f_t \leftarrow f_s$
end for
$f_0(\mathbf{x}) = D_\theta\left(\{\mathbf{x}_i, f_t(\mathbf{x}_i)\}_{i\in\mathcal{C}}, t, \mathbf{x}\right)$

Figure 8: Functional diffusion algo. EQ 3,9,11 corresponding to EQ 49, 51, 53. (Image source: (Zhang and Wonka, 2023))

and our goal is to approximate $a(x), u(x)$. No matter what we do, we need to have two prior $D_\theta[u_t, t](x)$ and $D_\phi[a_t, t](x)$. And we set boundary condition and observations as condition data for

$D_\theta[u_t, t](x)$. Now the only thing left is that, we need to use function condition in either training or sampling.

7.1 Function condition in training

If we incorporate function condition in training, then our objective is a joint objective as follow,

$$\begin{aligned}\underset{\theta,\phi}{\operatorname{argmin}}\, w(t) \sum_{i\in\mathbf{Q}} \Big(&|\mathbf{D}_\theta\left[\{x_j, u_t(x_j)\}_{j\in\mathbf{C}}, t, x_i\right] - u_0(x_i)|^2 \\ &+ |\mathbf{D}_\phi\left[\{x_j, a_t(x_j)\}_{j\in\mathbf{C}}, t, x_i\right] - a_0(x_i)|^2 \\ &+ |\nabla_{x_i}\big(\mathbf{D}_\phi\left[\{x_j, a_t(x_j)\}_{j\in\mathbf{C}}, t, x_i\right] * \nabla_{x_i}\mathbf{D}_\theta\left[\{x_j, u_t(x_j)\}_{j\in\mathbf{C}}, t, x_i\right]\big) - f(x_i)|^2\Big)\end{aligned}$$

And sampling algorithm is unchanged. This is not exactly what we want, since we need to train θ, ϕ for any function condition.

7.2 Function condition in sampling

Another way is to incorporate function condition during sampling, similar to BlindDPS. Since our goal during sampling is just to predict $\{x_i, u_s(x_i), a_s(x_i)\}_{i\in\mathbf{C}}$ Suppose we get $\{x_i, u_s^{'}(x_i), a_s^{'}(x_i)\}_{i\in\mathbf{C}}$ from

$$u_s(x) = \frac{\sigma_s}{\sigma_t}u_t(x) + \Big(\alpha_s - \sigma_s\frac{\alpha_t}{\sigma_t}\Big)\mathbf{D}_\theta[\{x_i, u_t(x_i)\}_{i\in\mathbf{C}}, t, x] \tag{55}$$

$$a_s(x) = \frac{\sigma_s}{\sigma_t}a_t(x) + \Big(\alpha_s - \sigma_s\frac{\alpha_t}{\sigma_t}\Big)\mathbf{D}_\theta[\{x_i, a_t(x_i)\}_{i\in\mathbf{C}}, t, x] \tag{56}$$

then we need to 'correct' $\{x_i, u_s^{'}(x_i), a_s^{'}(x_i)\}_{i\in\mathbf{C}}$ with function condition.

$$\underset{a(x_j),u(x_j)}{\operatorname{argmin}} \sum_{i\in\mathbf{Q}} |\nabla_{x_i}\big(\mathbf{D}_\phi\left[\{x_j, a_t(x_j)\}_{j\in\mathbf{C}}, t, x_i\right] * \nabla_{x_i}\mathbf{D}_\theta\left[\{x_j, u_t(x_j)\}_{j\in\mathbf{C}}, t, x_i\right]\big) - f(x_i)|^2 \tag{57}$$

7.3 Use function condition implicitly during training

The 3D shape experiment in (Zhang and Wonka, 2023), finding $f(x)$ is equivalent to solve Eikonal equation. However, they never used function condition! And their final Eikonal metric $\mathbf{EIKONAL}(f) = \frac{1}{|\epsilon_x|}\sum_{i\in\epsilon_x}|||\nabla f(x_i)|||^2$ is around 0.024, with only 64 observation points. Boundary metric is around 0.012. I believed this is achieve because they trained denoiser so well, and function dataset is choose in a way that all function satisfy EIKONAL. So they learnt a good prior.

7.4 Our previous methods

Our problem setting could be write as,

$$\begin{aligned} p(u_0, a_0|BC, FC, OBS) &\propto p(BC, FC, OBS|u_0, a_0)p(u_0|a_0)p(a_0) && (58)\\ &\propto p(BC, OBS|u_0)p(FC|u_0, a_0)p(u_0|a_0)p(a_0) && (59)\end{aligned}$$

Suppose we have a pre-trained FNO or DeepONet, where $\mathbf{G}^\dagger a_0 \approx u_0$ for any $\{a_0, u_0\} \sim p(u_0, a_0)$. Then we can get rid of second and third term in rhs of eqs 59. And we can only perform diffusion on

a_0. And the posterior score is

$$\nabla_{a_k} \log p(\mathbf{G}^{\dagger}(\hat{a_0}), a_k | BC, FC, OBS) \approx \nabla_{a_k} \log p(a_k) + \nabla_{a_k} p(BC, OBS | \mathbf{G}^{\dagger}(\hat{a_0})) \tag{60}$$

Eqs(60) will be easy if we discretize a like an image, and set $\mathbf{G}^{\dagger}$ to a DeepONet.

References

Hyungjin Chung, Jeongsol Kim, Sehui Kim, and Jong Chul Ye. Parallel diffusion models of operator and image for blind inverse problems, 2022.

Hyungjin Chung, Jeongsol Kim, Michael T. Mccann, Marc L. Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems, 2023.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020.

Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. Journal of Machine Learning Research, 6(24):695–709, 2005. URL `http://jmlr.org/papers/v6/hyvarinen05a.html`.

Bahjat Kawar, Michael Elad, Stefano Ermon, and Jiaming Song. Denoising diffusion restoration models, 2022.

Charles Laroche, Andrés Almansa, and Eva Coupete. Fast diffusion em: a diffusion model for blind inverse problems with application to deconvolution, 2023.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2022.

Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. Sliced score matching: A scalable approach to density and score estimation, 2019.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations, 2021.

Pascal Vincent. A connection between score matching and denoising autoencoders. Neural Computation, 23:1661–1674, 2011. URL `https://api.semanticscholar.org/CorpusID:5560643`.

Biao Zhang and Peter Wonka. Functional diffusion, 2023.