Diffusion Models

Saeid Naderiparizi

CPSC 532S

November 17, 2022

Generative models

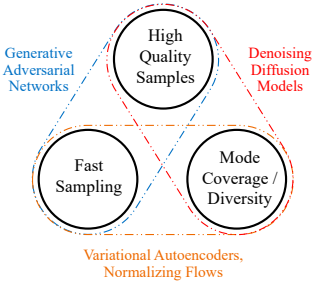
Learning distribution of data given a set of samples from it $\mathcal{D} \overset{\text{i.i.d.}}{\sim} q(x)$.

- Simple parameterized distributions e.g. $\mathcal{N}(\mu, \sigma)$
- Normalizing Flows
- Variational Auto-Encoder (VAE)s
- Generative Adversarial Network (GAN)s
- Diffusion Models
- ...

Generative models

Learning distribution of data given a set of samples from it $\mathcal{D} \overset{\text{i.i.d.}}{\sim} q(x)$.

- Simple parameterized distributions e.g. $\mathcal{N}(\mu, \sigma)$
- Normalizing Flows
- Variational Auto-Encoder (VAE)s
- Generative Adversarial Network (GAN)s
- Diffusion Models
- ...



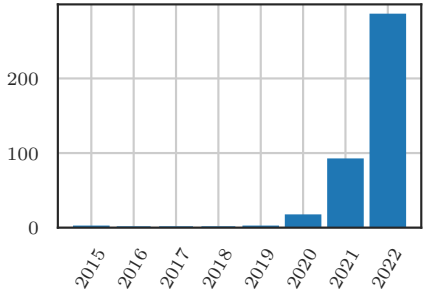
(Xiao et al., 2021)

Diffusion Models in 2022

Papers submitted to arXiv under CS category with "diffusion" in their title, *only in 2022* (as of November 16, 2022) [link]

Diffusion Models over time

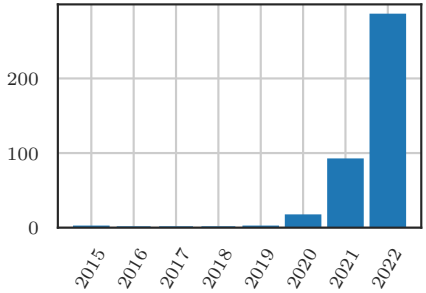
Some researchers from Oxford host a website that maintains a list of diffusion/score-based model papers at `https://scorebasedgenerativemodeling.github.io/`
Below shows the number of papers published in this area annually, according to this website:



Diffusion Models over time

Some researchers from Oxford host a website that maintains a list of diffusion/score-based model papers at `https://scorebasedgenerativemodeling.github.io/`
Below shows the number of papers published in this area annually, according to this website:

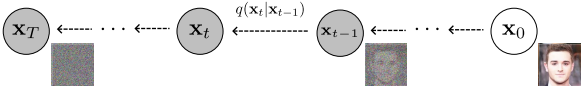
- Sohl-Dickstein et al. (2015) first introduced diffusion models in their current format.
- (Ho et al., 2020) proposed Denoising Diffusion Probabilistic Models (DDPM).



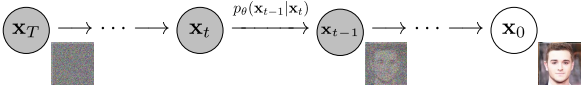
Overview

Diffusion Models (Overview)

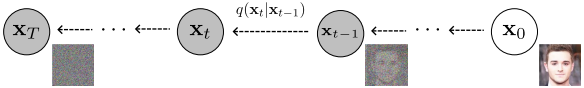
- Forward process: creating pure noise from the data by slowly adding noise to it.



- Reverse Process: creating data from noise *by inverting the forward process.*



Forward Process



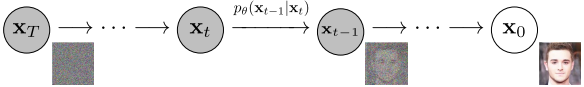
(Ho et al., 2020)

Consider a sequence of noise scales $0 < \beta_1, \beta_2, \ldots, \beta_N < 1$. Let $\mathbf{x}_0 \sim \mathcal{D}$ be a data point. A Markov chain is constructed such that

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) := \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}), \qquad q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\,\mathbf{x}_{t-1}, \beta_t \boldsymbol{I}).$$

The noise scales are prescribed such that approximately $q(\mathbf{x}_T|\mathbf{x}_0) \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ for any $\mathbf{x}_0$.

Reverse Process



(Ho et al., 2020)

$$p_\theta(\mathbf{x}_{0:T}) := p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t), \quad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)).$$

- $p(\mathbf{x}_T) = \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
- When the noise levels β_t are small enough, Gaussian conditionals in the reverse Markov chain ensure enough expressivity.
- To ensure $q(\mathbf{x}_T|q(\mathbf{x}_0)) \approx \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, we require many diffusion steps T.

Training

Training objective is to minimize the negative log-likelihood (NLL) of the data:

$$\theta^* = \arg\min_{\theta} \mathbb{E}_{\mathbf{x}\sim\mathcal{D}} \left[-\log p_\theta(\mathbf{x})\right]$$

Training

Training objective is to minimize the negative log-likelihood (NLL) of the data:

$$\theta^* = \arg\min_{\theta} \mathbb{E}_{\mathbf{x}\sim\mathcal{D}} \left[-\log p_\theta(\mathbf{x}) \right]$$

In diffusion models we instead optimize the Evidence Lower Bound (ELBO):

$$\log p_\theta(\mathbf{x}_0) \geq \mathbb{E}_q \left[\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \right] = \mathbb{E}_q \left[\log p(\mathbf{x}_T) + \sum_{t\geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})} \right].$$

Training

Training objective is to minimize the negative log-likelihood (NLL) of the data:

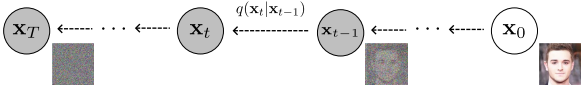
$$\theta^* = \arg\min_{\theta} \mathbb{E}_{\mathbf{x}\sim\mathcal{D}} \left[-\log p_\theta(\mathbf{x}) \right]$$

In diffusion models we instead optimize the Evidence Lower Bound (ELBO):

$$\log p_\theta(\mathbf{x}_0) \geq \mathbb{E}_q \left[\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \right] = \mathbb{E}_q \left[\log p(\mathbf{x}_T) + \sum_{t\geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})} \right].$$

Challenge: this objective is expensive to compute. Can we sub-sample t?

Properties of the Forward Process



(Ho et al., 2020)

Because all the distributions in the forward process are Gaussian, we can analytically compute various marginals and posteriors (Sohl-Dickstein et al., 2015). In particular,

- Timestep skipping:

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\alpha_t}\mathbf{x}_0, (1-\alpha_t)\boldsymbol{I}), \qquad \alpha_t := \prod_{j=1}^{t}(1-\beta_j)$$

- Posterior:

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t\boldsymbol{I}),$$

$$\text{where} \quad \tilde{\boldsymbol{\mu}}(\mathbf{x}_t, \mathbf{x}_0) := \frac{\sqrt{\alpha_{t-1}}\beta_t}{1-\alpha_t}\mathbf{x}_0 + \frac{\sqrt{1-\beta_t}(1-\alpha_{t-1})}{1-\alpha_t}\mathbf{x}_t \quad \text{and} \quad \tilde{\beta}_t := \frac{1-\alpha_{t-1}}{1-\alpha_t}\beta_t.$$

Making training feasible

Sohl-Dickstein et al. (2015) proposed the following objective function

$$
\begin{aligned}
L &= \mathbb{E}_q \left[\log p(\mathbf{x}_T) + \sum_{t \geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})} \right] \\
&= \mathbb{E}_q \Big[\underbrace{\mathrm{KL}\left(q(\mathbf{x}_T|\mathbf{x}_0)||p(\mathbf{x}_T)\right)}_{L_T} + \sum_{t>1} \underbrace{\mathrm{KL}\left(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)\right)}_{L_{t-1}} \underbrace{- \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0} \Big].
\end{aligned}
$$

Training: L_T

$$L = \mathbb{E}_q \Big[\underbrace{\mathrm{KL}\left(q(\mathbf{x}_T|\mathbf{x}_0)||p(\mathbf{x}_T)\right)}_{L_T} + \sum_{t>1} \underbrace{\mathrm{KL}\left(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)\right)}_{L_{t-1}} \underbrace{- \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0} \Big].$$

This term is constant.

Training: L_0

$$L = \mathbb{E}_q \Big[\underbrace{\mathrm{KL}\left(q(\mathbf{x}_T|\mathbf{x}_0)||p(\mathbf{x}_T)\right)}_{L_T} + \sum_{t>1} \underbrace{\mathrm{KL}\left(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)\right)}_{L_{t-1}} \underbrace{- \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0} \Big].$$

This term is a likelihood term corresponding to the last step of the reverse process. In practice, it is usually a Gaussian distribution or a discretized Gaussian distribution.

Training: $L_{1:T-1}$

$$L = \mathbb{E}_q \Big[\underbrace{\mathrm{KL}\left(q(\mathbf{x}_T|\mathbf{x}_0)||p(\mathbf{x}_T)\right)}_{L_T} + \sum_{t>1} \underbrace{\mathrm{KL}\left(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)\right)}_{L_{t-1}} \underbrace{- \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0} \Big].$$

These terms are KL divergences between two Gaussians, hence are analytically computable.

- Remember that $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$.
- Further, let $\boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t) = \sigma_t^2 \boldsymbol{I}$ where σ_t^2 is a fixed hyperparameter. Then,

$$L_{t-1} = \mathbb{E}_q \left[\frac{1}{2\sigma_t^2} \left\| \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_t, t) \right\|^2 \right] + C$$

- In practice, $\sigma_t^2 = \beta_t$ (from $q(\mathbf{x}_t|\mathbf{x}_{t-1})$) or $\sigma_t^2 = \tilde{\beta}_t$ (from $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$) works well (Ho et al., 2020).

Re-parameterizations of the reverse process

$$L_{t-1} = \mathbb{E}_q \left[\frac{1}{2\sigma_t^2} \left\| \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_t, t) \right\|^2 \right] + C$$

Recall that $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\alpha_t}\mathbf{x}_0, (1-\alpha_t)\boldsymbol{I})$. Therefore,

$$\mathbf{x}_t = \sqrt{\alpha_t} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}).$$

Then we can rewrite $\tilde{\boldsymbol{\mu}}(\mathbf{x}_t, \mathbf{x}_0)$ as

$$\tilde{\boldsymbol{\mu}}(\mathbf{x}_t, \mathbf{x}_0) = \frac{1}{\sqrt{1-\beta_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}} \boldsymbol{\epsilon} \right).$$

Ho et al. (2020) proposed a different parameterization of the reverse process in which

$$\boldsymbol{\mu}_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{1-\beta_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right).$$

Re-parameterizations of the reverse process

Using this re-parameterization of the reverse process, we can rewrite L_{t-1} as

$$L_{t-1} = \mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}} \left[\underbrace{\frac{\beta_t^2}{2\sigma_t^2(1-\beta_t)(1-\alpha_t)}}_{\gamma_t} \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right\|^2 \right] + C.$$

Re-weighting the training objective

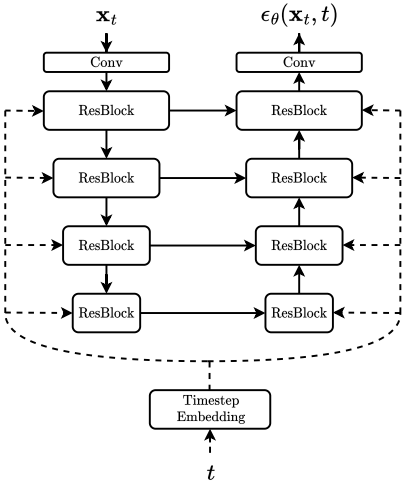
$$L_{t-1} = \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\underbrace{\frac{\beta_t^2}{2\sigma_t^2(1-\beta_t)(1-\alpha_t)}}_{\gamma_t}\left\|\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_\theta(\mathbf{x}_t,t)\right\|^2\right]+C.$$

The particular coefficients γ_t ensure that the training objective is weighted properly for the maximum data likelihood training. However, γ_t is often very large for smaller t's.
Ho et al. (2020) observed that the following objective derived by simply setting $\gamma_t = 1$ leads to higher image quality:

$$L_{\text{simple}} := \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\left\|\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_\theta(\mathbf{x}_t,t)\right\|^2\right].$$

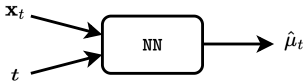
U-Net

- For image data, Ho et al. (2020) proposed using a particular type of U-Net architecture (Ronneberger et al., 2015) as the architecture for ϵ_θ.
- In this U-Net model shown on the right, there are self-attention layers added to a few of the lower-resolution ResBlocks.

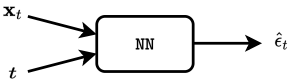


Different parameterizations of the reverse process

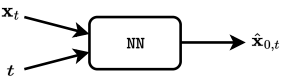
- $\hat{\boldsymbol{\mu}}_t$



- $\hat{\boldsymbol{\epsilon}}_t$

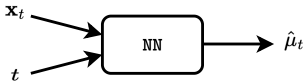


- $\hat{\mathbf{x}}_{0,t}$



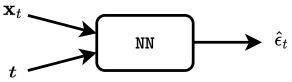
Different parameterizations of the reverse process

- $\hat{\boldsymbol{\mu}}_t$

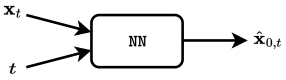


$$\mathbf{x}_{t-1} \sim \mathcal{N}(\hat{\boldsymbol{\mu}}_t, \sigma_t \boldsymbol{I}).$$

- $\hat{\boldsymbol{\epsilon}}_t$

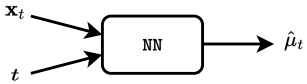


- $\hat{\mathbf{x}}_{0,t}$



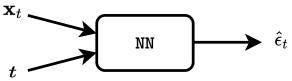
Different parameterizations of the reverse process

- $\hat{\boldsymbol{\mu}}_t$



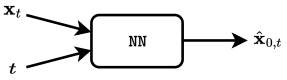
$$\mathbf{x}_{t-1} \sim \mathcal{N}(\hat{\boldsymbol{\mu}}_t, \sigma_t \boldsymbol{I}).$$

- $\hat{\boldsymbol{\epsilon}}_t$



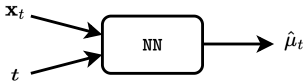
$$\hat{\boldsymbol{\mu}}_t = \frac{1}{\sqrt{1-\beta_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}} \hat{\boldsymbol{\epsilon}}_t \right), \text{ then } \mathbf{x}_{t-1} \sim \mathcal{N}(\hat{\boldsymbol{\mu}}_t, \sigma_t \boldsymbol{I}).$$

- $\hat{\mathbf{x}}_{0,t}$



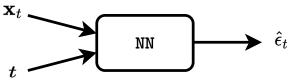
Different parameterizations of the reverse process

- $\hat{\boldsymbol{\mu}}_t$



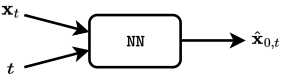
$$\mathbf{x}_{t-1} \sim \mathcal{N}(\hat{\boldsymbol{\mu}}_t, \sigma_t \boldsymbol{I}).$$

- $\hat{\boldsymbol{\epsilon}}_t$



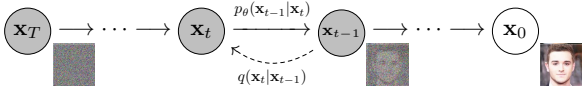
$$\hat{\boldsymbol{\mu}}_t = \frac{1}{\sqrt{1-\beta_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}} \hat{\boldsymbol{\epsilon}}_t \right), \text{ then } \mathbf{x}_{t-1} \sim \mathcal{N}(\hat{\boldsymbol{\mu}}_t, \sigma_t \boldsymbol{I}).$$

- $\hat{\mathbf{x}}_{0,t}$



$$\hat{\boldsymbol{\mu}}_t = \tilde{\boldsymbol{\mu}}(\mathbf{x}_t, \hat{\mathbf{x}}_{0,t}), \text{ then } \mathbf{x}_{t-1} \sim \mathcal{N}(\hat{\boldsymbol{\mu}}_t, \sigma_t \boldsymbol{I}).$$

DDPM Recap



Forward Process

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) := \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}),$$

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\,\mathbf{x}_{t-1}, \beta_t \boldsymbol{I})$$

Reverse Process

$$p_\theta(\mathbf{x}_{0:T}) := p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t),$$

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

Reverse process parameterization

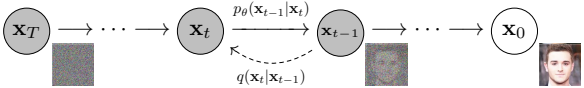
$$\boldsymbol{\mu}_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{1-\beta_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right)$$

$$\boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t) = \sigma_t^2 \boldsymbol{I}$$

(Simple) objective function

$$\mathbb{E}_{t, \mathbf{x}_0, \boldsymbol{\epsilon}} \left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\|^2 \right]$$

DDPM Recap



Algorithm 1 Training

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

Algorithm 2 Sampling

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

Forward Process as Stochastic Differential Equation

Forward process: $q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \boldsymbol{I})$.

Consider the limit of infinitely many small steps:

$$
\begin{aligned}
\mathbf{x}_t &= \sqrt{1-\beta_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\mathcal{N}(\mathbf{0}, \boldsymbol{I}) \\
&= \sqrt{1-\beta(t)\Delta t}\mathbf{x}_{t-1} + \sqrt{\beta(t)\Delta t}\mathcal{N}(\mathbf{0}, \boldsymbol{I}), \quad (\beta_t := \beta(t)\Delta t) \\
&\approx \mathbf{x}_{t-1} - \frac{\beta(t)\Delta t}{2}\mathbf{x}_{t-1} + \sqrt{\beta(t)\Delta t}\mathcal{N}(\mathbf{0}, \boldsymbol{I}), \quad \text{(Taylor expansion)} \\
\rightarrow d\mathbf{x}_t &= \underbrace{-\frac{1}{2}\beta(t)\mathbf{x}_t dt}_{\text{drift term}} + \underbrace{\sqrt{\beta(t)}d\mathbf{w}}_{\text{diffusion term}}
\end{aligned}
$$

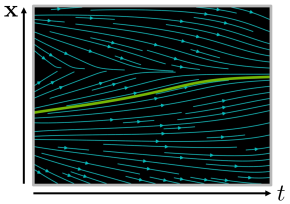
It is a special case of the more general SDE formulation used in generative diffusion models (Song et al., 2020b)

$$
d\mathbf{x}_t = \underbrace{f(t)\mathbf{x}_t dt}_{\text{drift term}} + \underbrace{g(t)d\mathbf{w}}_{\text{diffusion term}} \quad .
$$

Differential Equations

Ordinary Differential Equation (ODE)

$$d\mathbf{x} = f(\mathbf{x}, t)dt$$



Solution:

$$\mathbf{x}(t) = \mathbf{x}(0) + \int_0^t f(\mathbf{x}, \tau)d\tau$$

Numerical solution:

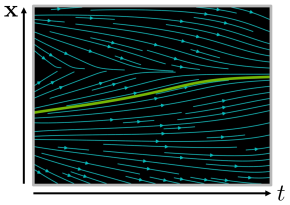
$$\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + f(\mathbf{x}(t), t)\Delta t$$

https://cvpr2022-tutorial-diffusion-models.github.io/

Differential Equations

Ordinary Differential Equation (ODE)

$$d\mathbf{x} = f(\mathbf{x}, t)dt$$



Solution:

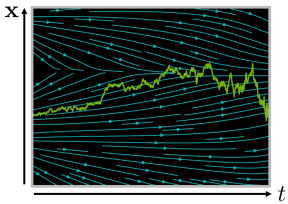
$$\mathbf{x}(t) = \mathbf{x}(0) + \int_0^t f(\mathbf{x}, \tau)d\tau$$

Numerical solution:

$$\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + f(\mathbf{x}(t), t)\Delta t$$

Stochastic Differential Equation (SDE)

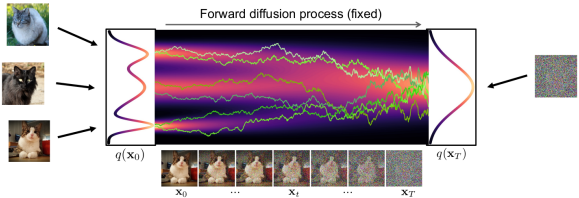
$$d\mathbf{x} = f(\mathbf{x}, t)dt + g(\mathbf{x}, t)d\mathbf{w}$$



$$\mathbf{x}(t+\Delta t) \approx \mathbf{x}(t)+f(\mathbf{x}(t), t)\Delta t+g(\mathbf{x}(t), t)\sqrt{\Delta t}\mathcal{N}(\mathbf{0}, \boldsymbol{I})$$

https://cvpr2022-tutorial-diffusion-models.github.io/

Forward Process as Stochastic Differential Equation



https://cvpr2022-tutorial-diffusion-models.github.io/

SDEs of the form $d\mathbf{x}_t = f(t)\mathbf{x}_t dt + g(t)d\mathbf{w}$ admit closed-from $q_t(\mathbf{x}_t|\mathbf{x}_0)$. For example, for

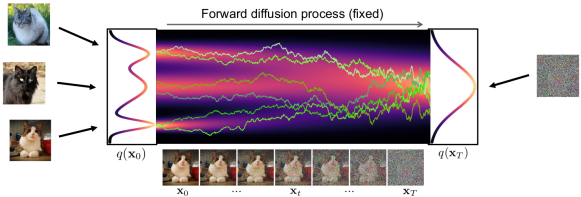
$$d\mathbf{x}_t = -\frac{1}{2}\beta(t)\mathbf{x}_t dt + \sqrt{\beta(t)}d\mathbf{w},$$

we have

$$q_t(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \alpha_t\mathbf{x}_0, \sigma_t^2\boldsymbol{I})$$

$$\alpha_t = \exp\left(-\frac{1}{2}\int_0^t \beta(s)ds\right) \qquad \sigma_t^2 = 1 - \exp\left(-\int_0^t \beta(s)ds\right)$$

Reverse Process as Stochastic Differential Equation



According to Anderson (1982), for a **forward-time SDE** of the form

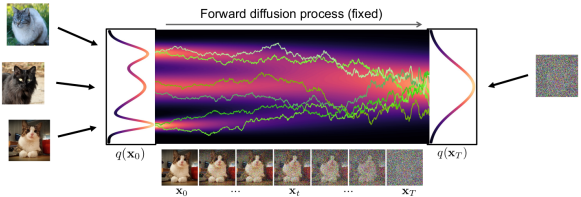
$$d\mathbf{x}_t = f(t)\mathbf{x}_t dt + g(t)d\mathbf{w}$$

the **reverse-time SDE** is

$$d\mathbf{x}_t = \underbrace{\left[f(t)\mathbf{x}_t - g(t)^2 \overbrace{\nabla_{\mathbf{x}} \log q_t(\mathbf{x}_t)}^{\text{Score Function}}\right] dt}_{\text{drift term}} + \underbrace{g(t)d\bar{\mathbf{w}}}_{\text{diffusion term}} \quad ,$$

which can be solved using any numerical method for SDEs.

Reverse Process as Stochastic Differential Equation



According to Anderson (1982), for a **forward-time SDE** of the form

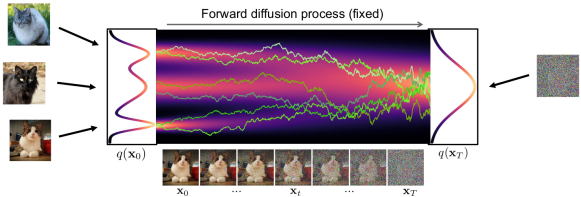
$$d\mathbf{x}_t = f(t)\mathbf{x}_t dt + g(t)d\mathbf{w}$$

the **reverse-time SDE** is

$$d\mathbf{x}_t = \underbrace{\Big[f(t)\mathbf{x}_t - g(t)^2 \overbrace{\nabla_{\mathbf{x}} \log q_t(\mathbf{x}_t)}^{\text{Score Function}}\Big] dt}_{\text{drift term}} + \underbrace{g(t)d\bar{\mathbf{w}}}_{\text{diffusion term}} \quad ,$$

which can be solved using any numerical method for SDEs.
Q: How to estimate the score function?

Reverse Process as Stochastic Differential Equation



According to Anderson (1982), for a **forward-time SDE** of the form

$$d\mathbf{x}_t = f(t)\mathbf{x}_t dt + g(t)d\mathbf{w}$$

the **reverse-time SDE** is

$$d\mathbf{x}_t = \underbrace{\Big[f(t)\mathbf{x}_t - g(t)^2 \overbrace{\nabla_{\mathbf{x}} \log q_t(\mathbf{x}_t)}^{\text{Score Function}}\Big] dt}_{\text{drift term}} + \underbrace{g(t)d\bar{\mathbf{w}}}_{\text{diffusion term}} \quad ,$$

which can be solved using any numerical method for SDEs.

Q: How to estimate the score function? A: Score matching

Score Matching

Train a network $s_\theta(\mathbf{x}_t, t)$ to estimate $\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t)$.

$$\min_\theta \mathbb{E}_{t\sim U(0,T)} \left[\mathbb{E}_{\mathbf{x}_0 \sim \mathcal{D}} \left[\mathbb{E}_{\mathbf{x}_t \sim q_t(\mathbf{x}_t|\mathbf{x}_0)} \left[\| s_\theta(\mathbf{x}_t, t) - \nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t) \|_2^2 \right] \right] \right]$$

But $\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t)$ is intractable!

Score Matching

Train a network $s_\theta(\mathbf{x}_t, t)$ to estimate $\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t)$.

$$\min_\theta \mathbb{E}_{t\sim U(0,T)} \left[\mathbb{E}_{\mathbf{x}_0\sim\mathcal{D}} \left[\mathbb{E}_{\mathbf{x}_t\sim q_t(\mathbf{x}_t|\mathbf{x}_0)} \left[\| s_\theta(\mathbf{x}_t, t) - \nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t) \|_2^2 \right] \right] \right]$$

But $\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t)$ is intractable!

Hyvärinen and Dayan (2005) proposed to equivalently optimize the following tractable objective

$$\min_\theta \mathbb{E}_{t\sim U(0,T)} \left[\mathbb{E}_{\mathbf{x}_0\sim\mathcal{D}} \left[\mathbb{E}_{\mathbf{x}_t\sim q_t(\mathbf{x}_t|\mathbf{x}_0)} \left[\frac{1}{2} \| s_\theta(\mathbf{x}_t, t) \|_2^2 + \mathrm{Tr}(\nabla_{\mathbf{x}} s_\theta(\mathbf{x}_t, t)) \right] \right] \right]$$

Denoising Score Matching

Train a network $s_\theta(\mathbf{x}_t, t)$ to estimate $\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t)$.

$$\min_\theta \mathbb{E}_{t\sim U(0,T)} \left[\mathbb{E}_{\mathbf{x}_0 \sim \mathcal{D}} \left[\mathbb{E}_{\mathbf{x}_t \sim q_t(\mathbf{x}_t|\mathbf{x}_0)} \left[\| s_\theta(\mathbf{x}_t, t) - \nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t) \|_2^2 \right] \right] \right]$$

But $\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t)$ is intractable!

Denoising Score Matching

Train a network $s_\theta(\mathbf{x}_t, t)$ to estimate $\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t)$.

$$\min_\theta \mathbb{E}_{t\sim U(0,T)} \left[\mathbb{E}_{\mathbf{x}_0\sim\mathcal{D}} \left[\mathbb{E}_{\mathbf{x}_t\sim q_t(\mathbf{x}_t|\mathbf{x}_0)} \left[\| s_\theta(\mathbf{x}_t, t) - \nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t) \|_2^2 \right] \right] \right]$$

But $\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t)$ is intractable!

Vincent (2011) proposed to instead optimize the following tractable objective

$$\min_\theta \mathbb{E}_{t\sim U(0,T)} \left[\mathbb{E}_{\mathbf{x}_0\sim\mathcal{D}} \left[\mathbb{E}_{\mathbf{x}_t\sim q_t(\mathbf{x}_t|\mathbf{x}_0)} \left[\| s_\theta(\mathbf{x}_t, t) - \nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t|\mathbf{x}_0) \|_2^2 \right] \right] \right],$$

which has the same optimal solution as the original objective i.e., $s_{\theta^*}(\mathbf{x}_t, t) = \nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t)$.

Denoising Score Matching (Reparameterziation)

Recall that $q_t(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \alpha_t \mathbf{x}_0, \sigma_t^2 \boldsymbol{I})$ for some α_t and σ_t functions. Considering the re-parameterization of this

$$\mathbf{x}_t = \alpha_t \mathbf{x}_0 + \sigma_t \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}),$$

We can derive the score function

$$\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t|\mathbf{x}_0) = \nabla_{\mathbf{x}_t} \left[-\frac{1}{2} \left(\frac{\mathbf{x}_t - \alpha_t \mathbf{x}_0}{\sigma_t} \right)^2 - \log(\sigma_t \sqrt{2\pi}) \right] = -\frac{\mathbf{x}_t - \alpha_t \mathbf{x}_0}{\sigma_t^2} = -\frac{\boldsymbol{\epsilon}}{\sigma_t}.$$

Therefore, the denoising score matching objective simplifies to

$$\min_\theta \mathbb{E}_{t \sim U(0,T)} \left[\mathbb{E}_{\mathbf{x}_0 \sim \mathcal{D}} \left[\mathbb{E}_{\mathbf{x}_t \sim q_t(\mathbf{x}_t|\mathbf{x}_0)} \left[\frac{1}{\sigma_t^2} \left\| \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) - \boldsymbol{\epsilon} \right\|_2^2 \right] \right] \right].$$

Denoising Score Matching (Loss weighting)

In practice, one can re-weight the loss terms for different target metrics.

$$\min_{\theta} \mathbb{E}_{t\sim U(0,T)} \left[\mathbb{E}_{\mathbf{x}_0\sim\mathcal{D}} \left[\mathbb{E}_{\mathbf{x}_t\sim q_t(\mathbf{x}_t|\mathbf{x}_0)} \left[\frac{\lambda(t)}{\sigma_t^2} \left\| \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) - \boldsymbol{\epsilon} \right\|_2^2 \right] \right] \right].$$

- Perceptual quality (FID, etc.): $\lambda(t) = \sigma_t^2$ $\longrightarrow L_{\text{simple}}$ objective
- Maximum log-likelihood: $\lambda(t) = \beta(t)$ $\longrightarrow -\text{ELBO}$

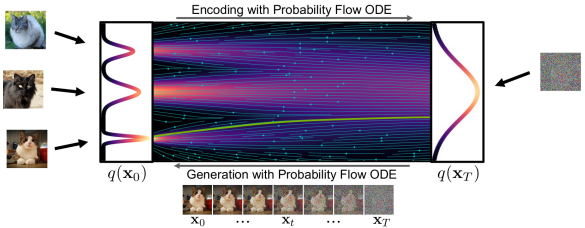
Probability flow ODE

Recall the reverse-time SDE

$$d\mathbf{x}_t = \Big[f(t)\mathbf{x}_t - g(t)^2\nabla_{\mathbf{x}}\log q_t(\mathbf{x}_t)\Big]dt + g(t)d\bar{\mathbf{w}}.$$

The following ODE is equivalent to this SDE in distribution (Song et al., 2020b):

$$d\mathbf{x}_t = \Big[f(t)\mathbf{x}_t - \frac{1}{2}g(t)^2\nabla_{\mathbf{x}}\log q_t(\mathbf{x}_t)\Big]dt.$$



Probability flow ODE

Recall the reverse-time SDE

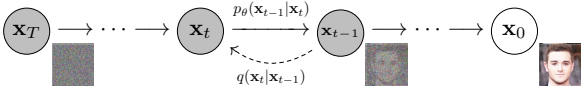
$$d\mathbf{x}_t = \Big[f(t)\mathbf{x}_t - g(t)^2 \nabla_{\mathbf{x}} \log q_t(\mathbf{x}_t)\Big] dt + g(t) d\bar{\mathbf{w}}.$$

The following ODE is equivalent to this SDE in distribution (Song et al., 2020b):

$$d\mathbf{x}_t = \Big[f(t)\mathbf{x}_t - \frac{1}{2} g(t)^2 \nabla_{\mathbf{x}} \log q_t(\mathbf{x}_t)\Big] dt.$$

- Deterministic forward and reverse processes
- Allows latent space interpolation
- Allows using advanced ODE solvers
- Allows evaluating the learned density
- Often slightly lower quality than SDE

Initial idea



Skip some steps in the reverse process.

- Predict $\hat{\mathbf{x}}_0$ from $\mathbf{x}_t$.
- $\mathbf{x}_{t'} \sim q(\mathbf{x}_{t'}|\mathbf{x}_t, \hat{\mathbf{x}}_0)$ for $t' < t-1$ which skips steps[1].

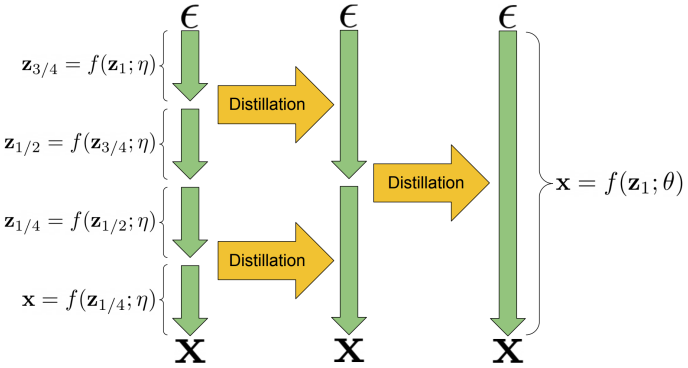
However, it leads to lower quality samples.

Song et al. (2020a) proposed DDIM which is a particular discritization of a probability flow ODE. DDIM produces higher quality samples after skipping steps.

[1] Note that $q(\mathbf{x}_{t'}|\mathbf{x}_t, \mathbf{x}_0)$ for any $0 < t' < t$ admits a closed form

Progressive Distillation

- Recursively distills the model into one with half the steps.
- It works on deterministic processes only (related to probability flow ODE)
- Can distill to 4 steps without losing much perceptual quality.

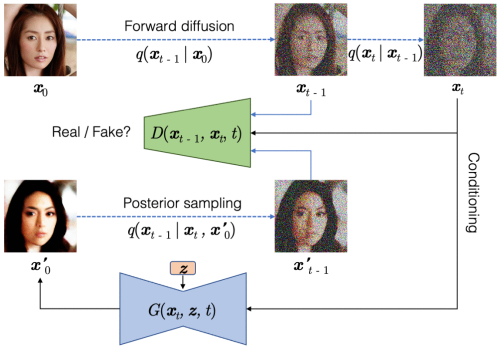


Denoising Diffusion GANs

(Xiao et al., 2021)

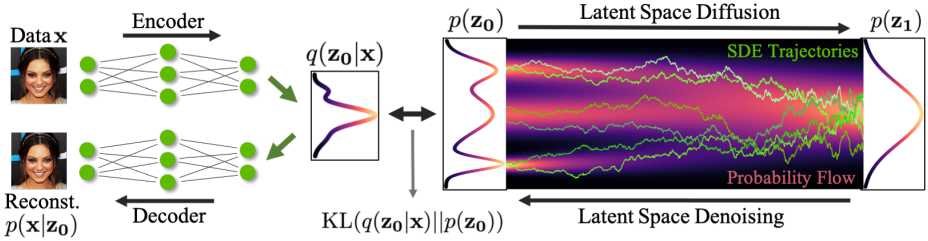
$$\min_{\theta} \sum_{t \geq 1} \mathbb{E}_{q(\mathbf{x}_t)} \left[D_{\text{adv}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)) \right]$$

With only 4 steps achieves comparable performance to DDPM with 1000 steps.



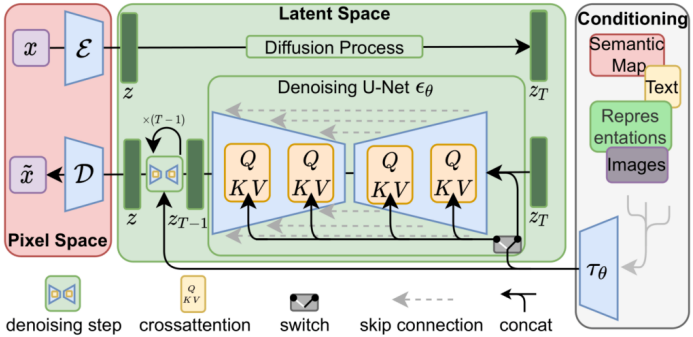
Diffusion Models in Latent Space

Diffusion Model + VAE (Vahdat et al., 2021)



Diffusion Models in Latent Space

Diffusion Model + VQVAE (Rombach et al., 2022)



Conditional Generation with Diffusion Models

So far, we focused on unconditional generation i.e., learning a distribution $p(\mathbf{x})$ given a dataset of i.i.d. samples from it $\mathcal{D} \overset{i.i.d.}{\sim} p(\mathbf{x})$.

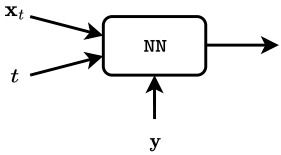
How can we generate conditional samples i.e., $p(\mathbf{x}|\mathbf{y})$ for some condition $\mathbf{y}$?

Examples:

- Class-conditional generation e.g., images of flamingos
- Image super-resolution
- Imputation e.g., image in-painting
- Colorization
- etc.

Conditional Diffusion Models

Conditional Diffusion Models are the most straight-forward way to generate conditional samples. If we have a dataset of $(\mathbf{x}, \mathbf{y})$ pairs, we can make the model learn $p(\mathbf{x}|\mathbf{y})$ by simply passing the conditions $\mathbf{y}$ as input to the network as well.



Controllable Generation

A unique and remarkable property of Diffusion models is their ability to sample from conditional distributions at test time, without re-training (Sohl-Dickstein et al., 2015; Song et al., 2020b).
It is referred to as "Classifier Guidance" too, but note that this method is
not limited to class-conditional generation.

Remember the connection of diffusion models to score functions.

Controllable Generation

A unique and remarkable property of Diffusion models is their ability to sample from conditional distributions at test time, without re-training (Sohl-Dickstein et al., 2015; Song et al., 2020b). It is referred to as "Classifier Guidance" too, but note that this method is not limited to class-conditional generation.

Remember the connection of diffusion models to score functions.

- Assume we have a pre-trained *unconditional* diffusion model s.t. $s_\theta(\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}_t} q_t(\mathbf{x}_t)$.
- All we need is to estimate $\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t|\mathbf{y})$

Controllable Generation

A unique and remarkable property of Diffusion models is their ability to sample from conditional distributions at test time, without re-training (Sohl-Dickstein et al., 2015; Song et al., 2020b). It is referred to as "Classifier Guidance" too, but note that this method is not limited to class-conditional generation.

Remember the connection of diffusion models to score functions.

- Assume we have a pre-trained *unconditional* diffusion model s.t. $s_\theta(\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}_t} q_t(\mathbf{x}_t)$.
- All we need is to estimate $\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t|\mathbf{y})$

$$\log q_t(\mathbf{x}_t|\mathbf{y}) = \log q_t(\mathbf{x}_t) + \log q(\mathbf{y}|\mathbf{x}_t) - \log q(\mathbf{y}) \qquad \text{(Bayes rule)}$$

Controllable Generation

A unique and remarkable property of Diffusion models is their ability to sample from conditional distributions at test time, without re-training (Sohl-Dickstein et al., 2015; Song et al., 2020b). It is referred to as "Classifier Guidance" too, but note that this method is not limited to class-conditional generation.

Remember the connection of diffusion models to score functions.

- Assume we have a pre-trained *unconditional* diffusion model s.t. $s_\theta(\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}_t} q_t(\mathbf{x}_t)$.
- All we need is to estimate $\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t|\mathbf{y})$

$$\log q_t(\mathbf{x}_t|\mathbf{y}) = \log q_t(\mathbf{x}_t) + \log q(\mathbf{y}|\mathbf{x}_t) - \log q(\mathbf{y}) \qquad \text{(Bayes rule)}$$

$$\Rightarrow \nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t|\mathbf{y}) = \nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log q(\mathbf{y}|\mathbf{x}_t) \approx \boxed{s_\theta(\mathbf{x}_t, t) + f(\mathbf{x}_t, t)}$$

Controllable Generation

A unique and remarkable property of Diffusion models is their ability to sample from conditional distributions at test time, without re-training (Sohl-Dickstein et al., 2015; Song et al., 2020b).
It is referred to as "Classifier Guidance" too, but note that this method is not limited to class-conditional generation.

Remember the connection of diffusion models to score functions.

- Assume we have a pre-trained *unconditional* diffusion model s.t. $s_\theta(\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}_t} q_t(\mathbf{x}_t)$.
- All we need is to estimate $\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t|\mathbf{y})$

$$\log q_t(\mathbf{x}_t|\mathbf{y}) = \log q_t(\mathbf{x}_t) + \log q(\mathbf{y}|\mathbf{x}_t) - \log q(\mathbf{y}) \qquad \text{(Bayes rule)}$$

$$\Rightarrow \nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t|\mathbf{y}) = \nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log q(\mathbf{y}|\mathbf{x}_t) \approx \boxed{s_\theta(\mathbf{x}_t, t) + f(\mathbf{x}_t, t)}$$

where $f(\mathbf{x}_t, t)$ is a discriminative model that predicts the label $\mathbf{y}$ from the noisy input $\mathbf{x}_t$.
In practice, a coefficient is added to the guidance term $s_\theta(\mathbf{x}_t, t) + w f(\mathbf{x}_t, t)$ to control faithfulness of the samples to the given condition.

Controllable Generation

A unique and remarkable property of Diffusion models is their ability to sample from conditional distributions at test time, without re-training (Sohl-Dickstein et al., 2015; Song et al., 2020b).
It is referred to as "Classifier Guidance" too, but note that this method is
not limited to class-conditional generation.

Remember the connection of diffusion models to score functions.

- Assume we have a pre-trained *unconditional* diffusion model s.t. $s_\theta(\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}_t} q_t(\mathbf{x}_t)$.
- All we need is to estimate $\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t|\mathbf{y})$

$$\log q_t(\mathbf{x}_t|\mathbf{y}) = \log q_t(\mathbf{x}_t) + \log q(\mathbf{y}|\mathbf{x}_t) - \log q(\mathbf{y}) \qquad \text{(Bayes rule)}$$

$$\Rightarrow \nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t|\mathbf{y}) = \nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log q(\mathbf{y}|\mathbf{x}_t) \approx \boxed{s_\theta(\mathbf{x}_t, t) + f(\mathbf{x}_t, t)}$$

where $f(\mathbf{x}_t, t)$ is a discriminative model that predicts the label $\mathbf{y}$ from the noisy input $\mathbf{x}_t$.
In practice, a coefficient is added to the guidance term $s_\theta(\mathbf{x}_t, t) + w f(\mathbf{x}_t, t)$ to control faithfulness of the samples to the given condition.

Example: in case of class-conditional generation, $f(\mathbf{x}_t, t)$ output of the softmax layer of a classifier trained on noisy images generated by the forward process.

Classifier-Free Guidance

Instead of training a separate classifier, we can implicitly get the required gradient via jointly training a conditional and unconditional model (Ho and Salimans, 2021).

$$\log q(\mathbf{y}|\mathbf{x}_t) = \log q(\mathbf{x}_t|\mathbf{y}) - \log q(\mathbf{x}_t) + \log q(\mathbf{y})$$
$$\Rightarrow \nabla_{\mathbf{x}_t} \log q(\mathbf{y}|\mathbf{x}_t) = \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t|\mathbf{y}) - \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t).$$

In practice, a single conditional diffusion model s_θ is trained that occasionally receives $\varnothing$ as conditioning input to represent unconditional generation.

$$\nabla_{\mathbf{x}_t} \log q(\mathbf{y}|\mathbf{x}_t) \approx s_\theta(\mathbf{x}_t, t, \mathbf{y}) - s_\theta(\mathbf{x}, t, \varnothing).$$

Like before, we can add the conditioning strength coefficient:

$$\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t) + w \nabla_{\mathbf{x}_t} \log q(\mathbf{y}|\mathbf{x}_t) \approx (1+w) s_\theta(\mathbf{x}_t, t, \mathbf{y}) - w s_\theta(\mathbf{x}_t, t, \varnothing).$$

Imputation

Ho and Salimans (2021) shows that an unconditional diffusion model can do data imputation (e.g. image in-painting) by simply replacing the observed part of the data with a noisy version of the observation in each timestep t.

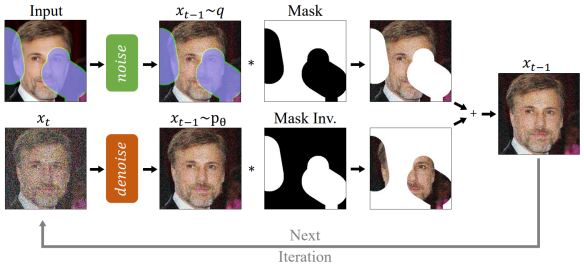


Image credit: (Lugmayr et al., 2022)

Image Super-Resolution

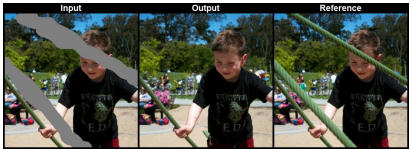
SR3 model: trains a conditional diffusion model that takes a lower-resolution image as input and learns to generate it in higher-resolution. (Saharia et al., 2022c)[1].

[1] https://iterative-refinement.github.io/

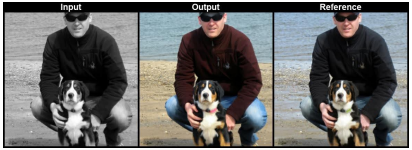
Image-to-image translation

Palette: large-scale conditional diffusion models trained on various image-to-image translation tasks (Saharia et al., 2022a)[1].

(a) Uncropping



(b) Inpainting

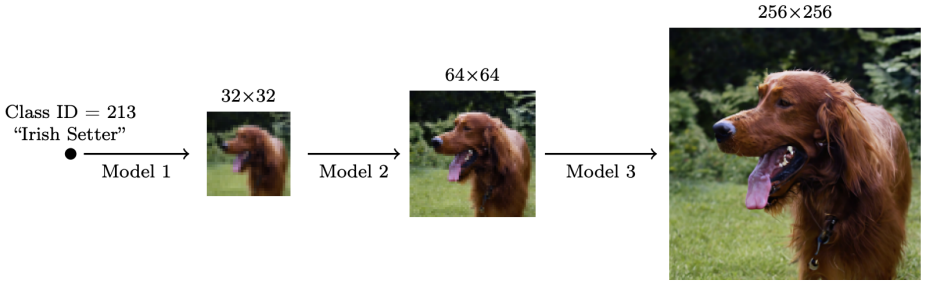


(c) Colorization

[1] https://iterative-refinement.github.io/palette/

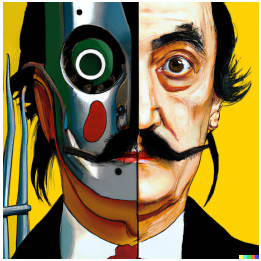
Cascaded Diffusion Models

(Ho et al., 2022b)[1].



[1] https://cascaded-diffusion.github.io/

Text-to-Image Generation



vibrant portrait painting of Salvador Dali with a robotic half face

a teddy bear on a skateboard in times square

DALL.E 2 (Ramesh et al., 2022)[1]

a cute sloth holding a small treasure chest. A bright golden glow is coming from the chest

a strawberry mug filled with white sesame seeds. The mug is floating in a dark chocolate sea

Imagen (Saharia et al., 2022b)[2]

[1] https://openai.com/dall-e-2/
[2] https://imagen.research.google/

Text-to-3D

DreamFusion: uses a pre-trained text-to-image model to learn a Neural Radial Fields (NeRF) modeling a 3D object (Poole et al., 2022)[1].

a baby bunny sitting on top of a stack of pancakes

Sydney opera house, aerial view

[1] https://dreamfusion3d.github.io/

Video Modeling

VDM (Ho et al., 2022c)[1]:

- 3D convolutional layers
- Cascaded Diffusion in spatial and temporal dimensions
- Conditioned on text
- Novel controllable generation

Imagen Video (Ho et al., 2022a)[2]:

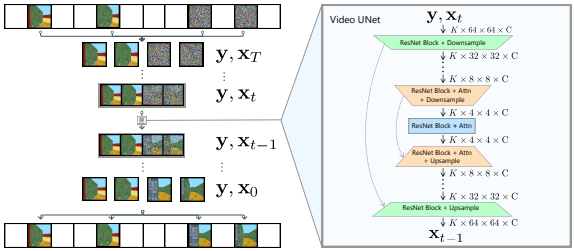
- Combines VDM with Imagen
- Trained on large-scale datasets

[1] https://video-diffusion.github.io/
[2] https://imagen.research.google/video/

Video Modeling

FDM (Harvey et al., 2022)[1]

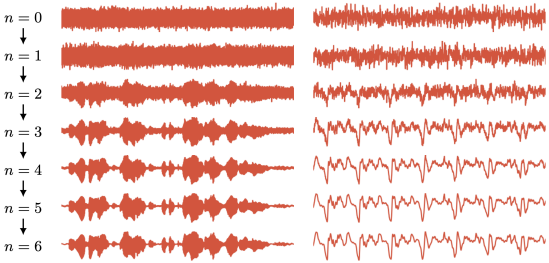
- Learns a single conditional diffusion model that is able to generate any subset of video frames conditioned on any other subset (as long as it fits in GPU memory).
- Can generate very long (1hour+) videos without loss of quality.



[1] https://plai.cs.ubc.ca/2022/05/20/flexible-diffusion-modeling-of-long-videos/

Speech synthesis

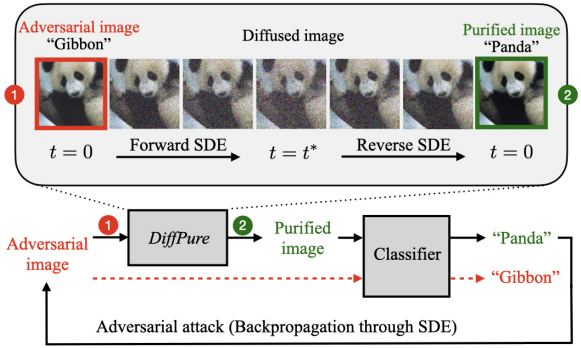
Wavegrad: applies diffusion models to waveform data. Since it is much lower-dimensional than images, it works with very few diffusion steps (Chen et al., 2020)[1].



[1] https://wavegrad.github.io/

Adversarial Purification

(Nie et al., 2022)[1].



[1] https://diffpure.github.io/

Additional resources

- Tutorial on diffusion models at CVPR 2022: `https://cvpr2022-tutorial-diffusion-models.github.io/`[1]
- Yang Song's blog post: `https://yang-song.net/blog/2021/score/`
- Lilian Weng's blog post: `https://lilianweng.github.io/posts/2021-07-11-diffusion-models/`
- Curated list of diffusion model papers
  - `https://scorebasedgenerativemodeling.github.io/`
  - `https://github.com/heejkoo/Awesome-Diffusion-Models`

[1] Many of the slides in today's talk were based on this tutorial.

References I

B. D. Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

N. Chen, Y. Zhang, H. Zen, R. J. Weiss, M. Norouzi, and W. Chan. Wavegrad: Estimating gradients for waveform generation. In *International Conference on Learning Representations*, 2020.

W. Harvey, S. Naderiparizi, V. Masrani, C. D. Weilbach, and F. Wood. Flexible diffusion modeling of long videos. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=0RTJcuvHtIu`.

J. Ho and T. Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.

J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

J. Ho, W. Chan, C. Saharia, J. Whang, R. Gao, A. Gritsenko, D. P. Kingma, B. Poole, M. Norouzi, D. J. Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022a.

J. Ho, C. Saharia, W. Chan, D. J. Fleet, M. Norouzi, and T. Salimans. Cascaded diffusion models for high fidelity image generation. *J. Mach. Learn. Res.*, 23:47–1, 2022b.

J. Ho, T. Salimans, A. Gritsenko, W. Chan, M. Norouzi, and D. J. Fleet. Video diffusion models. *arXiv preprint arXiv:2204.03458*, 2022c.

A. Hyvärinen and P. Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

References II

A. Lugmayr, M. Danelljan, A. Romero, F. Yu, R. Timofte, and L. Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11461–11471, 2022.

W. Nie, B. Guo, Y. Huang, C. Xiao, A. Vahdat, and A. Anandkumar. Diffusion models for adversarial purification. In *International Conference on Machine Learning (ICML)*, 2022.

B. Poole, A. Jain, J. T. Barron, and B. Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022.

A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

C. Saharia, W. Chan, H. Chang, C. Lee, J. Ho, T. Salimans, D. Fleet, and M. Norouzi. Palette: Image-to-image diffusion models. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–10, 2022a.

References III

C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. Denton, S. K. S. Ghasemipour, B. K. Ayan, S. S. Mahdavi, R. G. Lopes, et al. Photorealistic text-to-image diffusion models with deep language understanding. *arXiv preprint arXiv:2205.11487*, 2022b.

C. Saharia, J. Ho, W. Chan, T. Salimans, D. J. Fleet, and M. Norouzi. Image super-resolution via iterative refinement. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022c.

J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

J. Song, C. Meng, and S. Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2020a.

Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2020b.

A. Vahdat, K. Kreis, and J. Kautz. Score-based generative modeling in latent space. *Advances in Neural Information Processing Systems*, 34:11287–11302, 2021.

P. Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7): 1661–1674, 2011.

Z. Xiao, K. Kreis, and A. Vahdat. Tackling the generative learning trilemma with denoising diffusion gans. In *International Conference on Learning Representations*, 2021.