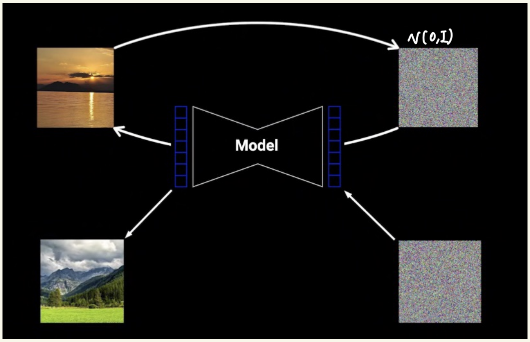
Diffusion Model

Diffusion Model

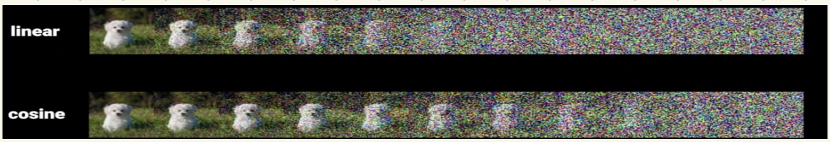
- The essential idea is to systematically and slowly destroy structure in a data distribution through an iterative forward diffusion process. We then learn a reverse diffusion process that restores structure in data, yielding a highly flexible and tractable generative model of the data.



- From "Outlier" YouTube

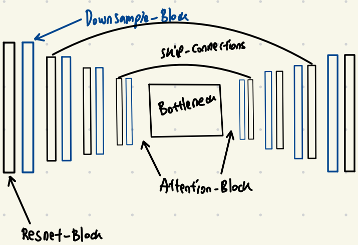
Forward Process : Applying noise to an image iteratively. Start off with the original image by adding more noise to image until the image become pure noise ($N(0,I)$) ★ Don't employ the same amount of noise at each time step. It's regulated by a schedule. It ensures variance doesn't explode.

linear & cosine schedule.



Reverse Process: Involves Neural Network learning to remove noise from an image step by step.

Architecture: U-Net Architecture



Model will always be designed for each time step and the way of telling which time step we are is done using the sinosidal embeddings

Notation

X_0 = Original image, X_T = Isotropic Gaussian

$q(X_t|X_{t-1})$: Forward Process

$p(X_{t-1}|X_t)$: Reverse Process

↳ p takes in an image X_t and produces a sample X_{t-1} using the Neural network.

Forward Process : $q(X_t|X_{t-1})$

$$q(X_t|X_{t-1}) = \mathcal{N}(\underset{\text{output}}{\underset{\uparrow}{X_t}}, \underset{\text{mean}}{\underset{\uparrow}{\sqrt{1-\beta_t}X_{t-1}}}, \underset{\uparrow}{\beta_t I})$$

variance, and β refers to the Schedule, and $\beta = [0,1]$

Linear schedule: $\beta_{start} = 0.0001, \beta_{end} = 0.02$. Grow linearly up to 0.02

Forward Process has a closed form Solution.

① let $\alpha_t = 1-\beta_t$

② $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$

③ By applying "reparametrization trick" ($\mathcal{N}(\mu, \sigma^2) = \mu + \sigma \cdot \varepsilon$), $q(X_t|X_{t-1}) = \mathcal{N}(X_t, \sqrt{1-\beta_t}X_{t-1}, \beta_t I) = \sqrt{1-\beta_t}X_{t-1} + \sqrt{\beta_t}\varepsilon$, where $\varepsilon \sim \mathcal{N}(0, I)$

④ Replace $1-\beta_t = \alpha_t$, then : $\sqrt{\alpha_t}X_{t-1} + \sqrt{1-\alpha_t}\varepsilon$

$$= \sqrt{\alpha_t\alpha_{t-1}}X_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\varepsilon$$

$$= \cdots = \sqrt{\alpha_t\alpha_{t-1}\cdots\alpha_1\alpha_0}X_0 + \sqrt{1-\alpha_t\alpha_{t-1}\cdots\alpha_1\alpha_0}\varepsilon$$

$$= \sqrt{\bar{\alpha}_t}X_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon = q(X_t|X_0) = \mathcal{N}(X_t ; \sqrt{\bar{\alpha}_t}X_0, (1-\bar{\alpha}_t)I)$$

Reverse Process : $p(X_{t-1}|X_t)$

fixed : We don't need a neural network

$$p(X_{t-1}|X_t) = \mathcal{N}(X_{t-1} ; \mu_\theta(X_t, t), \Sigma_\theta(X_t, t))$$

Two neural networks which parameterize the normal distribution

Loss function: $-\log(p_\theta(X_0))$: Negative log likelihood

However, $p_\theta(X_0)$ isn't tractable, hence we need ELBO.

$$-\log p_\theta(X_0) \leq -\log(p_\theta(X_0)) + D_{KL}(q(X_{1:T}|X_0) \,||\, p_\theta(X_{1:T}|X_0))$$

Diffusion model ~ VAE.

$$D_{KL}(q(X_{1:T}|X_0)) = \log\left(\frac{q(X_{1:T}|X_0)}{p_\theta(X_{1:T}|X_0)}\right) = \log\left(\frac{q(X_{1:T}|X_0)}{\frac{p_\theta(X_{0:T})}{p_\theta(X_0)}}\right) = \log\left(\frac{q(X_{1:T}|X_0)}{p_\theta(X_{0:T})}\right) + \log(p_\theta(X_0))$$

Bayes' Rule

$$\frac{p_\theta(X_0|X_{1:T})\,p_\theta(X_{1:T})}{p_\theta(X_0)} = \frac{p_\theta(X_0, X_{1:T})}{p_\theta(X_0)} = \frac{p_\theta(X_{0:T})}{p_\theta(X_0)}$$

Hence, ELBO is

$$-\log(p_\theta(x_0)) \leq \cancel{-\log(p_\theta(x_0))} + \log\left(\frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})}\right) \cancel{+\log(p_\theta(x_0))}$$

$$\Rightarrow -\log(p_\theta(x_0)) \leq \log\left(\frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})}\right) = \log\left(\frac{\prod_{t=1}^{T} q(x_t|x_{t-1})}{p(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)}\right) =$$

And, $p_\theta(x_{0:T})) = p(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)$

Algorithm 1: Training

repeat

$X_0 \sim q(X_0)$: Sample some image from our dataset

$t \sim \text{Uniform}(\{1, \dots, T\})$: Sample t

$\epsilon \sim \mathcal{N}(0, I)$

Take gradient descent step on

$$\nabla_\theta \| \epsilon - \epsilon_\theta (\sqrt{\bar{\alpha}_t} X_0 + \sqrt{1 - \bar{\alpha}_t}\, \epsilon, t) \|^2$$

Until converged

Algorithm 2: Sampling

$X_T \sim \mathcal{N}(0, I)$

for $t = T, \dots, 1$ do

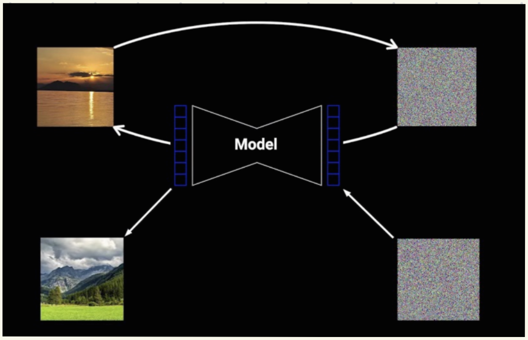
$z \sim \mathcal{N}(0, I)$ if $t > 1$, else $z = 0$

$$X_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(X_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta (X_t, t) \right) + \sigma_t z$$

end for

return X_0

Diffusion model: Systematically destroy data distribution through an iterative forward diffusion process, then learn a reverse diffusion process that restores structure in data

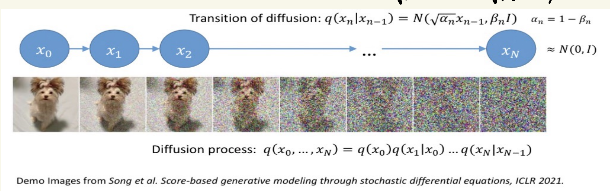


: From YouTube "Outlier"

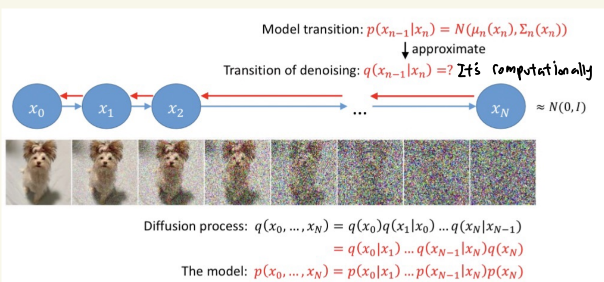
- Diffusion process gradually injects noise to data.
  - ↳ Described by a Markov Chain : $q(x_0, \dots, x_N) = q(x_0)q(x_1|x_0) \cdots q(x_N|x_{N-1})$

    where $q(x_t|x_{t-1}) = \mathcal{N}(x_t | \sqrt{1-\beta_t}\, x_{t-1}, \beta_t I)$

    Then $x_t = \sqrt{1-\beta_t}\, x_{t-1} + \sqrt{\beta_t}\, \varepsilon$, where $\varepsilon \sim \mathcal{N}(0, I)$



- Diffusion process in the reverse direction : Denoising process
  - ↳ Reverse factorization : $q(x_0, \dots, x_N) = q(x_0|x_1) \dots q(x_{N-1}|x_N)\, q(x_N)$



It's computationally intractable.

Notation

x_0 = Original image, x_T = Isotropic Gaussian

$q(x_t|x_{t-1})$: Forward Process

$p(x_{t-1}|x_t)$: Reverse Process

↳ p takes in an image x_t and produces a sample x_{t-1} using the Neural network.

Forward Process : $q(X_t|X_{t-1})$

$$q(X_t|X_{t-1}) = \mathcal{N}(\underset{\text{output}}{X_t}, \underset{\text{mean}}{\sqrt{1-\beta_t}X_{t-1}}, \underset{\text{variance}}{\beta_t I})$$

variance, and β refers to the Schedule, and $\beta = [0,1]$

Linear Schedule: $\beta_{start} = 0.0001$, $\beta_{end} = 0.02$. Grow linearly up to 0.02

Forward Process has a closed form Solution.

① let $a_t = 1-\beta_t$

② $\bar{a}_t = \prod_{s=1}^{t} a_s$

③ By applying "reparametrization trick" ($\mathcal{N}(\mu, \sigma^2) = \mu + \sigma \cdot \varepsilon$), $q(X_t|X_{t-1}) = \mathcal{N}(X_t, \sqrt{1-\beta_t}X_{t-1}, \beta_t I) = \sqrt{1-\beta_t}X_{t-1} + \sqrt{\beta_t}\,\varepsilon$, where $\varepsilon \sim \mathcal{N}(0, I)$

④ Replace $1-\beta_t = a_t$, then : $\sqrt{a_t}X_{t-1} + \sqrt{1-a_t}\,\varepsilon$

$$= \sqrt{a_t a_{t-1}}X_{t-2} + \sqrt{1-a_t a_{t-1}}\,\varepsilon$$

$$= \cdots = \sqrt{a_t a_{t-1} \cdots a_1 a_0}X_0 + \sqrt{1-a_t a_{t-1} \cdots a_1 a_0}\,\varepsilon$$

$$= \sqrt{\bar{a}_t}X_0 + \sqrt{1-\bar{a}_t}\,\varepsilon = q(X_t|X_0) = \mathcal{N}(X_t : \sqrt{\bar{a}_t}X_0, (1-\bar{a}_t)I)$$

For sampling : $X_t = \sqrt{\bar{a}_t}X_0 + \sqrt{(1-\bar{a}_t)}\,\varepsilon$ where $\varepsilon \sim \mathcal{N}(0, I)$

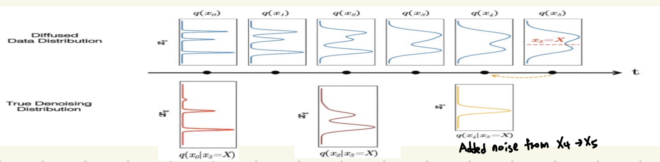
Reverse diffusion Process

- The goal of a diffusion model is to learn the reverse denoising process to iteratively undo the forward process
- Since $q(X_t|X_{t-1})$ is unknown (intractable), we try to approximate $q(X_t|X_{t-1})$ using P_θ function which is basically what neural network does

Why intractable?

$$q(X_t|X_{t-1}) = q(X_t|X_{t-1}) \cdot \frac{q(X_{t-1})}{q(X_t)}, \quad \text{where } q(X_t) = \int q(X_t|X_{t-1})\, q(X_{t-1})\, dx$$

↳ Computing this is computationally intractable



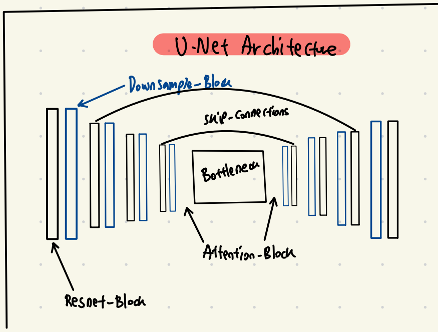
- The reverse process step $q(X_{t-1}|X_t)$ can be estimated as a Gaussian distribution too.

Hence, we can parameterize the learned reverse process as

$$P_\theta(X_{t-1}|X_t) = \mathcal{N}(X_{t-1}, \underbrace{\mu_\theta(X_t, t)}_{\text{Trainable network}}, \sigma_t^2 I)$$

Trainable network (U-Net, MLP, Denoising Autoencoder)

and, $P(X_T) = \mathcal{N}(\underset{\text{Output}}{X_T}, \underset{\mu}{0}, \underset{\sigma}{I})$



$: \underbrace{p_\theta(x_0|x_1)\, p_\theta(x_1|x_2) \cdots p_\theta(x_{T-1}|x_T)}_{\text{Markov Chain}} = \frac{p_\theta(x_0, x_1, \dots, x_T)}{p_\theta(x_1, \dots, x_{T-1}, x_T)} \cdot \frac{p_\theta(x_1, \dots, x_{T-1}, x_T)}{p_\theta(x_2, \dots, x_{T-1}, x_T)} \cdot \ \cdots \ \cdot \frac{p_\theta(x_{T-1}, x_T)}{p_\theta(x_T)}$

$$\Rightarrow p_\theta(x_{0:T}) = p(x_T) \cdot \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)$$

Now, how do we train diffusion model?

: We need to maximize the likelihood that an image we generate looks like it comes from the original data distribution.

We can bound the likelihood using ELBO (Variational Lower bound) just like VAE.

$L_{VLB} = L_T + L_{T-1} + \cdots + L_0$

where $L_T = D_{KL}(q(x_T|x_0) \,\|\, p_\theta(x_T))$

$L_t = D_{KL}(q(x_t|x_{t+1}, x_0) \,\|\, p_\theta(x_t|x_{t+1}))$ for $1 \le t \le T-1$

$L_0 = -\log p_\theta(x_0|x_1)$

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

where $\tilde{\mu}_t(x_t, x_0) := \frac{\sqrt{\bar{a}_{t-1}}\,\beta_t}{1-\bar{a}_t} x_0 + \frac{\sqrt{a_t}(1-\bar{a}_{t-1})}{1-\bar{a}_t} x_t$ and $\tilde{\beta}_t = \frac{1-\bar{a}_{t-1}}{1-\bar{a}_t}\beta_t$

And after doing some algebra, we get

$$L_{t-1} = \mathbb{E}_{x_0, \varepsilon}\left[\frac{1}{2\|\Sigma_\theta\|_2^2} \|\tilde{\mu}(x_t, x_0) - \mu_\theta(x_t, t)\|_2^2\right]$$

$$= \mathbb{E}_{x_0, \varepsilon}\left[\frac{1}{2\|\Sigma_\theta\|_2^2} \left\| \frac{1}{\sqrt{a_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{a}_t}}\varepsilon\right) - \mu_\theta(x_t, t)\right\|_2^2\right]$$

Instead of predicting μ, we should predict epsilon instead.

$$\tilde{\mu}_t(x_t, x_0) = \frac{1}{\sqrt{a_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{a}_t}}\varepsilon\right) \longrightarrow \mu_\theta(x_t, t) = \frac{1}{\sqrt{a_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{a}_t}}\varepsilon_\theta(x_t, t)\right)$$

Therefore, our loss becomes

$$L_{t-1} = \mathbb{E}_{x_0, \varepsilon}\left[\frac{\beta_t^2}{2a_t(1-\bar{a}_t)\|\Sigma_\theta\|_2^2} \left\|\varepsilon - \varepsilon_\theta\left(\sqrt{\bar{a}_t}\, x_0 + \sqrt{1-\bar{a}_t}\,\varepsilon, t\right)\right\|_2^2\right]$$

And the authors of DDPM says it's okay to drop all that baggage in front.

$$L_{t-1} = \mathbb{E}_{x_0, \varepsilon}\left[\left\|\varepsilon - \varepsilon_\theta\left(\sqrt{\bar{a}_t}\, x_0 + \sqrt{1-\bar{a}_t}\,\varepsilon, t\right)\right\|_2^2\right]$$

In Summary: Algorithm 1: Training

1: repeat

2: $x_0 \sim q(x_0)$

3: $t \sim \text{Uniform}(\{1, \dots, T\})$

4: $\varepsilon \sim \mathcal{N}(0, I)$

5: Take gradient descent step on

$\nabla_\theta \|\varepsilon - \varepsilon_\theta(\sqrt{\bar{a}_t}\, x_0 + \sqrt{1-\bar{a}_t}\,\varepsilon, t)\|^2$

6: until converged.

Algorithm 2 : Sampling (Inference)

1: $X_T \sim N(0, I)$

2: for $t = T, \dots, 1$ do

3: $\quad z \sim N(0, I)$ if $t > 1$, else $z = 0$

4: $\quad X_{t-1} = \frac{1}{\sqrt{a_t}}\left(X_t - \frac{1-a_t}{\sqrt{1-\bar{a}_t}}\varepsilon_\theta(X_t, t)\right) + \sigma_t z$

5: end for

6: return X_0

☆ $\sigma_t = \frac{1-\bar{a}_{t-1}}{1-\bar{a}_t}\beta_t$

Model will always be designed for each time step and the way of telling which time step we are is done using the sinosidal embeddings