Aprendizado Profundo

Diffusion Models

Lucas Silveira Kupssinskü

Machine Learning Theory and Applications Laboratory
PUCRS

agosto de 2026

Visão Geral

1. Modelos Geradores – Revisão
2. Encoder: Processo de Difusão
3. Decoder
4. Treinamento e Amostragem
5. Implementação e Extensões
6. Panorama

Visão Geral

Modelos Geradores

- O objetivo de um **modelo gerador** é aprender uma distribuição de probabilidade sobre uma variável **X** usando dados observados $\{x^{(i)}\}_{i=1}^{N}$.
- A densidade $p_X(x)$ é parametrizada por $\boldsymbol{\theta}$.
- Já vimos:
  - **GANs** (jogo min-max),
  - **Normalizing Flows** (transformações invertíveis),
  - **VAEs** (encoder/decoder + ELBO).

A próxima ideia: *Diffusion Models*[1]

- Definir um *encoder* **sem parâmetros treináveis** que degrada uma imagem adicionando ruído gaussiano de maneira sequencial.
- Treinar um *decoder* para **retirar o ruído** adicionado.

Intuição

Aprender a remover ruído um pequeno passo de cada vez é mais fácil que aprender a gerar uma imagem inteira do zero. Empilhando muitos passos pequenos, conseguimos transformar ruído puro em uma amostra realista.

[1] Jonathan Ho, Ajay Jain e Pieter Abbeel. "Denoising Diffusion Probabilistic Models". Em: NeurIPS. vol. 33. 2020, pp. 6840–6851.

Visão Geral

Visão Geral – Difusão

- No processo de difusão (*forward*) adicionamos ruído a cada passo até a imagem ficar indistinguível do ruído.
- No processo de *denoising* (decoder) treinamos uma rede neural para **remover um passo** de ruído.
- Após o treinamento podemos gerar novas instâncias partindo de $z_T \sim \mathcal{N}(0, \mathbf{I})$.

Difusão – Definição

O processo de difusão (*forward*) mapeia um exemplo x por uma série de etapas intermediárias $z_1, z_2, \ldots, z_T$ via adição de ruído:

$$z_1 = \sqrt{1 - \beta_1} \cdot x + \sqrt{\beta_1} \cdot \epsilon_1$$

$$z_t = \sqrt{1 - \beta_t} \cdot z_{t-1} + \sqrt{\beta_t} \cdot \epsilon_t \qquad \forall t \in \{2, 3, \ldots, T\}$$

- $\epsilon_t \sim \mathcal{N}(0, \mathbf{I})$: ruído amostrado da normal padrão.
- Hiperparâmetros $\beta_t \in [0, 1]$ são o ***noise schedule*** (controlam a velocidade da difusão).

Difusão – Forma probabilística (Markov)

Equivalentemente, podemos escrever:

$$q(z_1 \mid x) = \mathcal{N}_{z_1}\left[\sqrt{1-\beta_1} \cdot x,\ \beta_1 \cdot \mathbf{I}\right]$$

$$q(z_t \mid z_{t-1}) = \mathcal{N}_{z_t}\left[\sqrt{1-\beta_t} \cdot z_{t-1},\ \beta_t \cdot \mathbf{I}\right] \qquad \forall t \geq 2$$

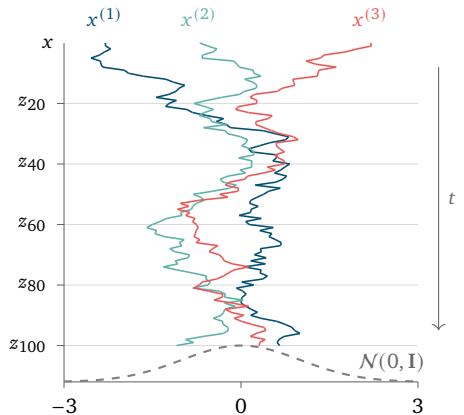
Cadeia de Markov

z_t depende apenas de z_{t-1}. Por isso, a distribuição conjunta fatoriza:

$$q(z_{1\ldots T} \mid x) = q(z_1 \mid x) \prod_{t=2}^{T} q(z_t \mid z_{t-1}).$$

Difusão – Trajetórias 1D

- Partimos de **três instâncias** $x^{(1)}, x^{(2)}, x^{(3)}$ (1D).
- Conforme os passos de difusão são aplicados, as instâncias são combinadas com ruído e vão aos poucos perdendo identidade.
- Após muitos passos, a distribuição converge para a distribuição do ruído $\mathcal{N}(0, \mathbf{I})$.



Trajetórias simuladas do *forward process* com $\beta = 0.04$ e $T = 100$.

Kernel de Difusão – Motivação

- Como o processo de difusão é probabilístico, para uma mesma instância x obtemos **várias amostras** z_t em cada instante.
- O processo definido até aqui é **sequencial**: para chegar em z_t precisamos passar por $z_1, \ldots, z_{t-1}$.
- Para acelerar, vamos derivar uma **forma fechada** para amostrar diretamente z_t a partir de x.

Kernel de Difusão – Substituindo z_1 em z_2

Partindo das definições:

$$z_1 = \sqrt{1-\beta_1} \cdot x + \sqrt{\beta_1} \cdot \epsilon_1, \qquad z_2 = \sqrt{1-\beta_2} \cdot z_1 + \sqrt{\beta_2} \cdot \epsilon_2.$$

Substituindo z_1:

$$z_2 = \sqrt{1-\beta_2}\left(\sqrt{1-\beta_1} \cdot x + \sqrt{\beta_1} \cdot \epsilon_1\right) + \sqrt{\beta_2} \cdot \epsilon_2.$$

Kernel de Difusão – Manipulação algébrica

Reescrevendo $\sqrt{\beta_1} = \sqrt{1 - (1 - \beta_1)}$:

$$z_2 = \sqrt{1 - \beta_2}\left(\sqrt{1 - \beta_1}\, x + \sqrt{1 - (1 - \beta_1)}\, \epsilon_1\right) + \sqrt{\beta_2}\, \epsilon_2.$$

Distribuindo:

$$z_2 = \sqrt{(1 - \beta_2)(1 - \beta_1)}\, x + \sqrt{(1 - \beta_2) - (1 - \beta_2)(1 - \beta_1)}\, \epsilon_1 + \sqrt{\beta_2}\, \epsilon_2.$$

Kernel de Difusão – Soma de gaussianas

Lembrete

A soma de duas gaussianas independentes com médias nulas e desvios σ_a, σ_b é uma gaussiana com média zero e variância $\sigma_a^2 + \sigma_b^2$.

Os dois últimos termos viram um único ruído gaussiano:

$$z_2 = \sqrt{(1-\beta_2)(1-\beta_1)} \cdot x + \sqrt{1-(1-\beta_2)(1-\beta_1)} \cdot \epsilon.$$

Observação

Podemos repetir esse processo para todos os z_t, sempre acumulando os fatores $(1-\beta_s)$.

Kernel de Difusão – Forma geral

Definindo $\alpha_t = \prod_{s=1}^{t}(1-\beta_s)$, temos:

$$\boxed{z_t = \sqrt{\alpha_t}\cdot x + \sqrt{1-\alpha_t}\cdot \epsilon} \qquad \epsilon \sim \mathcal{N}(0, \mathbf{I})$$

Em forma probabilística:

$$q(z_t \mid x) = \mathcal{N}_{z_t}\left[\sqrt{\alpha_t}\cdot x,\ (1-\alpha_t)\cdot \mathbf{I}\right].$$

Vantagem prática

Podemos amostrar z_t **diretamente** a partir de x sem simular toda a cadeia. Isso permite treinamento por *minibatch* eficiente.

Exemplo Numérico: *Schedule* β e Sinal vs Ruído

Pense em x como uma imagem limpa (e.g., um pixel ou um vetor latente). β_t é a variância de ruído adicionada no passo t do *forward process*; α_t é o quanto sobrou de x após t passos. Aqui usamos $T = 4$ para visualização, DDPM real tem $T = 1000$ e β na faixa 10^{-4} a 0.02, mas a curva é a mesma.

Schedule linear com $T = 4$: $\boldsymbol{\beta} = (0.1, 0.2, 0.3, 0.4)$. Acumulamos $\alpha_t = \prod_{s=1}^{t}(1 - \beta_s)$:

t	α_t	$\sqrt{\alpha_t}$ (sinal)	$\sqrt{1-\alpha_t}$ (ruído)
1	0.900	0.949	0.316
2	0.720	0.849	0.529
3	0.504	0.710	0.704
4	0.302	0.550	0.836

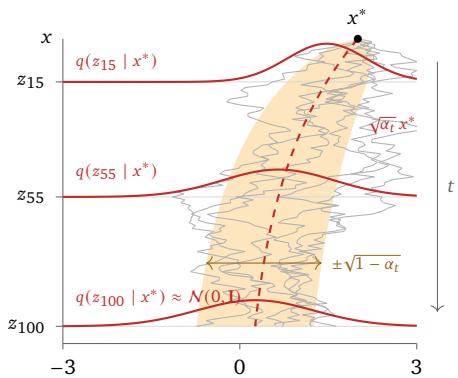
Ex.: $\alpha_3 = 0.9 \cdot 0.8 \cdot 0.7 = 0.504$.

Para $x = 1$:

$$z_t = \sqrt{\alpha_t} \cdot 1 + \sqrt{1 - \alpha_t} \cdot \epsilon$$

Em $t = 3$ o sinal e o ruído se cruzam; em $t = T$ grande, $\alpha_T \to 0$ e $z_T \approx \mathcal{N}(0, \mathbf{I})$ **por construção**, é o que permite o *decoder* começar a amostragem de ruído puro.

Kernel de Difusão – Visualização



Mesmas condições do slide de trajetórias ($\beta = 0.04$, $T = 100$), agora com todas as trajetórias partindo de $x^* = 2$.

O mesmo x^* gera várias trajetórias (em cinza). O kernel resume onde elas podem estar no instante t:

$$q(z_t \mid x) = \mathcal{N}_{z_t}\left[\sqrt{\alpha_t} \cdot x,\ (1-\alpha_t) \cdot \mathbf{I}\right]$$

- $\sqrt{\alpha_t}\, x$ (linha tracejada): a **média**. O sinal original encolhe geometricamente rumo a zero.
- $1 - \alpha_t$ (faixa): a **variância**. A incerteza acumulada pelo ruído cresce rumo a 1.
- Curvas sólidas: o kernel $q(z_t \mid x^*)$ em $t = 15, 55, 100$. Amostrar dele dispensa simular a cadeia passo a passo.
- Em $t = 100$: $\alpha_t \approx 0$, então o kernel já é aproximadamente $\mathcal{N}(0, \mathbf{I})$.

Visão Geral

Distribuição condicional $q(z_{t-1} \mid z_t, x)$

Para o decoder vamos precisar de uma expressão para a difusão **revertida no tempo**, condicionada em x:

$$q(z_{t-1} \mid z_t, x) = \frac{q(z_t \mid z_{t-1}, x)\, q(z_{t-1} \mid x)}{q(z_t \mid x)}$$

Por regra de Bayes. Vamos manipular essa expressão.

Distribuição condicional – Simplificação Markov

- Como difusão é Markov: $q(z_t \mid z_{t-1}, x) = q(z_t \mid z_{t-1})$.
- Tirando o denominador, a igualdade vira proporcionalidade:

$$q(z_{t-1} \mid z_t, x) \;\propto\; q(z_t \mid z_{t-1})\, q(z_{t-1} \mid x).$$

Distribuição condicional – Produto de gaussianas

Substituindo cada termo pela sua respectiva normal:

$$\propto \mathcal{N}_{z_t}\left[\sqrt{1-\beta_t}\cdot z_{t-1},\ \beta_t\,\mathbf{I}\right]\cdot \mathcal{N}_{z_{t-1}}\left[\sqrt{\alpha_{t-1}}\cdot x,\ (1-\alpha_{t-1})\,\mathbf{I}\right].$$

Mudança de variável

O primeiro fator é uma gaussiana em z_t, mas pode ser reescrito como uma gaussiana em z_{t-1}:

$$\mathcal{N}_{z_t}\left[\sqrt{1-\beta_t}\cdot z_{t-1},\ \beta_t\,\mathbf{I}\right] \ \propto\ \mathcal{N}_{z_{t-1}}\left[\frac{1}{\sqrt{1-\beta_t}}\,z_t,\ \frac{\beta_t}{1-\beta_t}\,\mathbf{I}\right].$$

O produto de duas gaussianas em z_{t-1} é proporcional a uma gaussiana em z_{t-1}; como o lado esquerdo é uma distribuição, a normalização é automática:

$$q(z_{t-1}\mid z_t, x) = \mathcal{N}_{z_{t-1}}\left[\frac{(1-\alpha_{t-1})}{1-\alpha_t}\sqrt{1-\beta_t}\,z_t + \frac{\sqrt{\alpha_{t-1}}\,\beta_t}{1-\alpha_t}\,x,\ \frac{\beta_t(1-\alpha_{t-1})}{1-\alpha_t}\,\mathbf{I}\right]$$

Decoder (*Reverse Process*)

O aprendizado está **todo no decoder**:

$$z_T \;\longrightarrow\; z_{T-1} \;\longrightarrow\; \cdots \;\longrightarrow\; z_1 \;\longrightarrow\; x.$$

- A distribuição reversa real $q(z_{t-1} \mid z_t)$ é complexa e potencialmente multimodal, depende dos dados.
- Na prática, **aproximamos** essa reversa por uma normal:

$$p(z_T) = \mathcal{N}_{z_T}[0, \mathbf{I}], \qquad p(z_{t-1} \mid z_t, \boldsymbol{\theta}_t) = \mathcal{N}_{z_{t-1}}\left[f_t[z_t, \boldsymbol{\theta}_t],\ \sigma_t^2\,\mathbf{I}\right],$$

$$p(x \mid z_1, \boldsymbol{\theta}_1) = \mathcal{N}_x\left[f_1[z_1, \boldsymbol{\theta}_1],\ \sigma_1^2\,\mathbf{I}\right].$$

Quando essa aproximação é razoável

Se o número de passos T for grande e β_t for próximo de 0, cada passo da reversa é quase determinístico e a aproximação gaussiana é boa.

Treinamento – O problema

Maximizar a log-verossimilhança dos dados é o objetivo:

$$\hat{\boldsymbol{\theta}}_{1\ldots T} = \underset{\boldsymbol{\theta}_{1\ldots T}}{\operatorname{argmax}} \left[\sum_{i=1}^{N} \log p(x^{(i)} \mid \boldsymbol{\theta}_{1\ldots T}) \right]$$

$$p(x \mid \boldsymbol{\theta}_{1\ldots T}) = \int p(x, z_{1\ldots T} \mid \boldsymbol{\theta}_{1\ldots T})\, dz_{1\ldots T}$$

Inviável

Essa integral é **intratável**. Vamos usar o **ELBO** como substituto.

Treinamento – Distribuições conjuntas

Conjunta do modelo (decoder)

$$p(x, z_{1\ldots T} \mid \boldsymbol{\theta}_{1\ldots T}) = p(x \mid z_1, \boldsymbol{\theta}_1) \prod_{t=2}^{T} p(z_{t-1} \mid z_t, \boldsymbol{\theta}_t) \cdot p(z_T)$$

Conjunta da difusão (encoder)

$$q(z_{1\ldots T} \mid x) = q(z_1 \mid x) \prod_{t=2}^{T} q(z_t \mid z_{t-1})$$

Tudo gaussiano $\Rightarrow$ tudo tratável passo a passo.

Evidence Lower Bound (ELBO)

Multiplicamos e dividimos por $q(z_{1\ldots T} \mid x)$, e aplicamos a desigualdade de Jensen:

$$\begin{aligned}
\log p(x \mid \boldsymbol{\theta}_{1\ldots T}) &= \log\left[\int p(x, z_{1\ldots T} \mid \boldsymbol{\theta}_{1\ldots T})\, dz\right] \\
&= \log\left[\int q(z_{1\ldots T} \mid x) \frac{p(x, z_{1\ldots T} \mid \boldsymbol{\theta}_{1\ldots T})}{q(z_{1\ldots T} \mid x)}\, dz\right] \\
&\geq \int q(z_{1\ldots T} \mid x) \log\left[\frac{p(x, z_{1\ldots T} \mid \boldsymbol{\theta}_{1\ldots T})}{q(z_{1\ldots T} \mid x)}\right] dz \quad \text{(Jensen)}
\end{aligned}$$

Vamos trabalhar nesse termo (o ELBO).

ELBO – Substituindo as conjuntas

Substituindo as definições do encoder e decoder:

$$\log\left[\frac{p(x|z_1,\boldsymbol{\theta}_1)\prod_{t=2}^{T}p(z_{t-1}|z_t,\boldsymbol{\theta}_t)\cdot p(z_T)}{q(z_1|x)\prod_{t=2}^{T}q(z_t|z_{t-1})}\right]$$

$$=\log\left[\frac{p(x|z_1,\boldsymbol{\theta}_1)}{q(z_1|x)}\right]+\log\left[\frac{\prod_{t=2}^{T}p(z_{t-1}|z_t,\boldsymbol{\theta}_t)}{\prod_{t=2}^{T}q(z_t|z_{t-1})}\right]+\log p(z_T).$$

ELBO – Aplicando Bayes em $q(z_t|z_{t-1})$

Pela regra de Bayes (e Markov):

$$q(z_t \mid z_{t-1}) = \frac{q(z_{t-1} \mid z_t, x)\, q(z_t \mid x)}{q(z_{t-1} \mid x)}.$$

Aplicando isso na razão dentro do produtório, vários termos se cancelam, sobrando apenas $q(z_1 \mid x)$ no numerador e $q(z_T \mid x)$ no denominador:

$$\prod_{t=2}^{T} \frac{1}{q(z_t \mid z_{t-1})} = \prod_{t=2}^{T} \frac{q(z_{t-1} \mid x)}{q(z_{t-1} \mid z_t, x)\, q(z_t \mid x)}.$$

ELBO – O cancelamento em detalhe

Separamos o produtório em uma parte que permanece e outra cujos termos se cancelam:

$$\prod_{t=2}^{T} \frac{q(z_{t-1} \mid x)}{q(z_{t-1} \mid z_t, x)\, q(z_t \mid x)} = \underbrace{\prod_{t=2}^{T} \frac{1}{q(z_{t-1} \mid z_t, x)}}_{\text{permanece}} \cdot \underbrace{\prod_{t=2}^{T} \frac{q(z_{t-1} \mid x)}{q(z_t \mid x)}}_{\text{cancela}}$$

Ao expandir essa segunda parte, cada denominador cancela o numerador seguinte, sobrando apenas os extremos:

$$\prod_{t=2}^{T} \frac{q(z_{t-1} \mid x)}{q(z_t \mid x)} = \frac{q(z_1 \mid x)}{\cancel{q(z_2 \mid x)}} \cdot \frac{\cancel{q(z_2 \mid x)}}{\cancel{q(z_3 \mid x)}} \cdots \frac{\cancel{q(z_{T-1} \mid x)}}{q(z_T \mid x)} = \frac{q(z_1 \mid x)}{q(z_T \mid x)}.$$

ELBO – Simplificação

Após o cancelamento, a expressão fica:

$$= \log\left[\frac{p(x|z_1,\boldsymbol{\theta}_1)\, q(z_1|x)}{q(z_1|x)}\right] + \log\left[\frac{\prod_{t=2}^{T} p(z_{t-1}|z_t,\boldsymbol{\theta}_t)}{\prod_{t=2}^{T} q(z_{t-1}|z_t,x)}\right] + \log\left[\frac{p(z_T)}{q(z_T|x)}\right]$$

$$= \log p(x|z_1,\boldsymbol{\theta}_1) + \log\left[\frac{\prod_{t=2}^{T} p(z_{t-1}|z_t,\boldsymbol{\theta}_t)}{\prod_{t=2}^{T} q(z_{t-1}|z_t,x)}\right] + \cancelto{\approx 0}{\log\left[\frac{p(z_T)}{q(z_T|x)}\right]}$$

O último termo é desprezível: $q(z_T|x) \approx p(z_T)$ para T grande.

ELBO – Formulação final

Tomando esperança em $q(z_{1\ldots T} \mid x)$:

$$\text{ELBO}[\boldsymbol{\theta}_{1\ldots T}] = \mathbb{E}_{q(z_1|x)}\big[\log p(x|z_1, \boldsymbol{\theta}_1)\big] - \sum_{t=2}^{T} \mathbb{E}_{q(z_t|x)}\Big[D_{KL}\big[q(z_{t-1}|z_t, x) \,\|\, p(z_{t-1}|z_t, \boldsymbol{\theta}_t)\big]\Big]$$

- **Termo de reconstrução**: $\log p(x|z_1, \boldsymbol{\theta}_1)$ é uma normal $\Rightarrow$ MSE.
- **Termos KL**: ambas distribuições são normais $\Rightarrow$ **forma fechada**!

ELBO – KL entre normais

A KL entre duas gaussianas tem expressão fechada. Aplicando-a:

KL desenvolvida

$$\mathbb{E}_{q(z_t|x)}\left[D_{KL}(q \,\|\, p)\right] = \frac{1}{2\sigma_t^2}\left|\frac{1-\alpha_{t-1}}{1-\alpha_t}\sqrt{1-\beta_t}\, z_t + \frac{\sqrt{\alpha_{t-1}}\,\beta_t}{1-\alpha_t}\, x - f_t[z_t, \boldsymbol{\theta}_t]\right|^2$$

Substituindo na expressão do ELBO obtemos a **Diffusion Loss Function**.

Diffusion Loss Function

$$\mathcal{L}[\boldsymbol{\theta}_{1\ldots T}] = \underbrace{\sum_{i=1}^{N}\left(-\log \mathcal{N}_x[f_1[z_1,\boldsymbol{\theta}_1],\sigma_1^2\mathbf{I}]\right)}_{\text{reconstrução}} + \underbrace{\sum_{t=2}^{T}\frac{1}{2\sigma_t^2}\left|\frac{1-\alpha_{t-1}}{1-\alpha_t}\sqrt{1-\beta_t}\,z_t + \frac{\sqrt{\alpha_{t-1}}\beta_t}{1-\alpha_t}\,x - f_t[z_t,\boldsymbol{\theta}_t]\right|^2}_{\text{erro quadrático no } \textit{timestep } t}$$

- Forma de **MSE** em vez de log-verossimilhança intratável.
- Cada termo t pede a rede f_t a prever a média da reversa em z_{t-1}.

Porém... um detalhe empírico

Observação experimental

Após treinar diversos modelos difusores que tentavam descobrir diretamente a imagem, percebeu-se empiricamente que os resultados eram **melhores** ao usar a rede para **estimar o ruído** ϵ adicionado, e não a imagem.

Vamos reparametrizar a função de custo para que a rede prediga ϵ em vez de z_{t-1}.

Reparametrizando a loss – Isolar x

Do kernel de difusão $z_t = \sqrt{\alpha_t}\, x + \sqrt{1-\alpha_t}\, \epsilon$:

$$x = \frac{z_t}{\sqrt{\alpha_t}} - \frac{\sqrt{1-\alpha_t}}{\sqrt{\alpha_t}}\, \epsilon.$$

Substituímos essa expressão de x dentro da loss e simplificamos.

Reparametrizando a loss – Agrupando z_t

O argumento dentro do quadrado (termo KL) era:

$$\frac{1-\alpha_{t-1}}{1-\alpha_t}\sqrt{1-\beta_t}\,z_t + \frac{\sqrt{\alpha_{t-1}}\,\beta_t}{1-\alpha_t}\,x - f_t[z_t, \boldsymbol{\theta}_t].$$

Substituímos $x = \frac{z_t}{\sqrt{\alpha_t}} - \frac{\sqrt{1-\alpha_t}}{\sqrt{\alpha_t}}\,\epsilon$ e usamos $\sqrt{\alpha_t} = \sqrt{1-\beta_t}\,\sqrt{\alpha_{t-1}}$.

Coeficiente de z_t

$$\frac{(1-\alpha_{t-1})\sqrt{1-\beta_t}}{1-\alpha_t} + \frac{\beta_t}{(1-\alpha_t)\sqrt{1-\beta_t}} = \frac{\overbrace{(1-\alpha_{t-1})(1-\beta_t)+\beta_t}^{=1-\alpha_t}}{(1-\alpha_t)\sqrt{1-\beta_t}} = \frac{1}{\sqrt{1-\beta_t}},$$

pois $\alpha_{t-1}(1-\beta_t) = \alpha_t$.

Reparametrizando a loss – Resultado

Coeficiente de ϵ

$$-\frac{\sqrt{\alpha_{t-1}}\,\beta_t}{1-\alpha_t}\cdot\frac{\sqrt{1-\alpha_t}}{\sqrt{\alpha_t}} = -\frac{\beta_t}{\sqrt{1-\alpha_t}\,\sqrt{1-\beta_t}}, \qquad \text{pois } \frac{\sqrt{\alpha_{t-1}}}{\sqrt{\alpha_t}} = \frac{1}{\sqrt{1-\beta_t}}.$$

Logo, o argumento se reduz a:

$$\frac{1}{\sqrt{1-\beta_t}}\,z_t - \frac{\beta_t}{\sqrt{1-\alpha_t}\sqrt{1-\beta_t}}\,\epsilon - f_t[z_t, \boldsymbol{\theta}_t].$$

Reparametrização da rede

Definimos f_t na mesma forma, trocando ϵ pela **nova rede g_t que prediz o ruído**:

$$f_t[z_t, \boldsymbol{\theta}_t] = \frac{1}{\sqrt{1-\beta_t}}\,z_t - \frac{\beta_t}{\sqrt{1-\alpha_t}\sqrt{1-\beta_t}}\,g_t[z_t, \boldsymbol{\theta}_t].$$

Reparametrizando a rede – Cancelamento

Como f_t tem a **mesma forma** do argumento, ao substituí-la os termos em z_t (coeficiente $\frac{1}{\sqrt{1-\beta_t}}$ idêntico) **se cancelam**:

$$\cancel{\frac{1}{\sqrt{1-\beta_t}} z_t} - \frac{\beta_t}{\sqrt{1-\alpha_t}\sqrt{1-\beta_t}} \epsilon - \cancel{\frac{1}{\sqrt{1-\beta_t}} z_t} + \frac{\beta_t}{\sqrt{1-\alpha_t}\sqrt{1-\beta_t}} g_t[z_t, \boldsymbol{\theta}_t].$$

Sobra apenas a diferença entre ruído predito e real, escalada por um fator:

$$\frac{\beta_t}{\sqrt{1-\alpha_t}\sqrt{1-\beta_t}} \left(g_t[z_t, \boldsymbol{\theta}_t] - \epsilon\right).$$

Reparametrizando a rede – Loss final

Substituindo, o termo do quadrado vira simplesmente $|g_t[z_t, \boldsymbol{\theta}_t] - \epsilon_t|^2$ a menos de uma constante multiplicativa, que podemos absorver durante a otimização:

$$\mathcal{L}[\boldsymbol{\theta}_{1\ldots T}] = \sum_{i=1}^{N} \sum_{t=2}^{T} \left| g_t[z_t, \boldsymbol{\theta}_t] - \epsilon_t \right|^2$$

Função de custo do modelo difusor

É um erro quadrático médio entre o ruído real ϵ_t adicionado em cada passo e a estimativa g_t da rede. **Surpreendentemente simples**.

O termo de reconstrução ($t = 1$) tem a mesma forma de MSE em ϵ; na prática o somatório roda de $t = 1$ a T.

Reparametrizando a rede – O que a rede prevê?

A rede pode ser treinada para prever alvos **equivalentes**; a escolha afeta a estabilidade e a qualidade:

- ϵ**-prediction** (DDPM, usado aqui): estima o ruído ϵ. Loss MSE simples e estável; é o default.
- x_0**-prediction**: estima diretamente a imagem limpa x. Equivalente pela álgebra do kernel, mas empiricamente pior em timesteps de ruído alto.
- v**-prediction** (Salimans & Ho, 2022): estima $v = \sqrt{\alpha_t}\,\epsilon - \sqrt{1-\alpha_t}\,x$, combinação de ruído e sinal. Estável em todo o schedule; comum em geração de vídeo e em distillation.

Resumo

As três são equivalentes via o kernel $z_t = \sqrt{\alpha_t}\,x + \sqrt{1-\alpha_t}\,\epsilon$; diferem na escala efetiva da loss por timestep.

Visão Geral

Algoritmo de Treinamento

Entrada: dados de treino x
Saída: parâmetros do modelo $\boldsymbol{\theta}_t$
repeat
  for $i \in \mathcal{B}$ **do** ▷ para cada exemplo do *minibatch*
    $t \sim \text{Uniform}[1, \ldots, T]$ ▷ sorteia o passo
    $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ ▷ sorteia o ruído
    $\ell_i = \left\| g_t\left[\sqrt{\alpha_t}\, x_i + \sqrt{1-\alpha_t}\,\epsilon,\ \boldsymbol{\theta}_t\right] - \epsilon \right\|^2$
  end for
  Acumule as *losses* do *batch* e dê um passo de gradiente
until convergir

Monta-se z_t pelo kernel $(\sqrt{\alpha_t}\, x_i + \sqrt{1-\alpha_t}\,\epsilon)$ e treina-se g_t para prever o ruído ϵ.

Algoritmo de Amostragem

Entrada: modelo $g_t[\,\cdot\,, \boldsymbol{\theta}_t]$
Saída: amostra x

$z_T \sim \mathcal{N}(0, \mathbf{I})$ ▹ última variável latente

for $t = T, \ldots, 2$ **do**

$$\hat{z}_{t-1} = \frac{1}{\sqrt{1-\beta_t}}\, z_t - \frac{\beta_t}{\sqrt{1-\alpha_t}\sqrt{1-\beta_t}}\, g_t[z_t, \boldsymbol{\theta}_t]$$ ▹ prediz o latente anterior

$\epsilon \sim \mathcal{N}(0, \mathbf{I})$ ▹ novo ruído

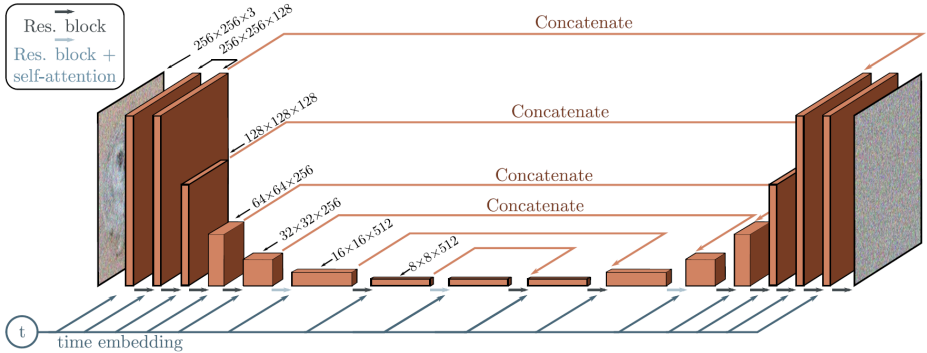
$z_{t-1} = \hat{z}_{t-1} + \sigma_t\, \epsilon$ ▹ adiciona ruído

end for

$$x = \frac{1}{\sqrt{1-\beta_1}}\, z_1 - \frac{\beta_1}{\sqrt{1-\alpha_1}\sqrt{1-\beta_1}}\, g_1[z_1, \boldsymbol{\theta}_1]$$ ▹ gera x sem ruído

Visão Geral

Modelo para estimar ϵ_t – *U-Net*



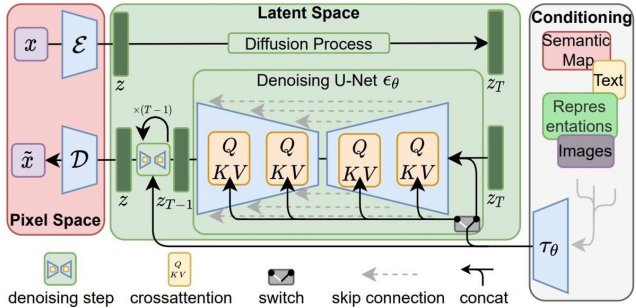
- Arquitetura padrão: **U-Net** com blocos residuais e auto-atenção.
- Recebe z_t + um time embedding de t, devolve uma estimativa de ϵ com a mesma forma de z_t.
- *Skip connections* preservam alta-frequência entre escalas.

Classifier-Free Diffusion Guidance[2]

- É comum adicionar uma **informação condicionante** no processo de geração.
- Pode parecer contraintuitivo, contudo informações sobre a classe que queremos gerar **auxiliam o processo** e geram imagens com maior qualidade.
- O treinamento condicionado é feito adicionando algum tipo de **embedding** da classe à U-Net.
  - Por exemplo, embeddings do BERT sobre a descrição da imagem (texto-para-imagem).
- Classifier-Free: durante o treino, esporadicamente "descondicionamos" a rede; na geração, interpolamos entre versões condicionada e incondicional para controlar a força da condição.

[2] Jonathan Ho e Tim Salimans. "Classifier-Free Diffusion Guidance". Em: arXiv preprint arXiv:2207.12598 (2022).

Latent Diffusion Models[3]



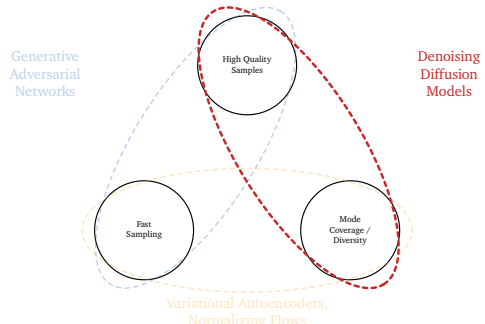
- Aplique a difusão no **espaço latente** de um VAE, não em pixels.
- Reduz drasticamente o custo computacional (latente é $\sim 64\times64$ em vez de 1024×1024).
- Base de *Stable Diffusion*.

[3] Robin Rombach et al. "High-Resolution Image Synthesis with Latent Diffusion Models". Em: CVPR. 2022, pp. 10684–10695.

Visão Geral

Onde modelos difusores ficam no trilema?

- **Qualidade visual**: ✓✓ (estado-da-arte em imagens).
- **Cobertura / diversidade**: ✓ (treino por NLL / ELBO cobre a distribuição toda).
- **Amostragem rápida**: × (tipicamente $T = 1000$ passos pela U-Net; técnicas como DDIM e consistency models atacam esse problema).



Posicionamento

Modelos difusores ocupam o canto qualidade + cobertura do trilema. Para baixa latência, *Latent Diffusion* e amostradores rápidos são o padrão.

Resumo

- **Difusão** (encoder, sem parâmetros): cadeia de Markov gaussiana que destrói gradualmente a estrutura de x.
- **Kernel**: forma fechada para amostrar z_t diretamente de x.
- **Reverse process** (decoder): rede neural aprende a reverter um passo da difusão.
- **ELBO**: termo de reconstrução + soma de KLs entre normais (todas com forma fechada).
- **Reparametrização**: prever ruído ϵ em vez da imagem leva a uma loss MSE simples.
- **Arquitetura**: U-Net com *time embedding*.
- **Extensões**: *classifier-free guidance*, *latent diffusion* (Stable Diffusion).

Leituras Recomendadas

- Capítulo 18: Simon J. D. Prince. Understanding Deep Learning. MIT Press, 2023
- Jonathan Ho, Ajay Jain e Pieter Abbeel. “Denoising Diffusion Probabilistic Models”. Em: NeurIPS. vol. 33. 2020, pp. 6840–6851
- Jonathan Ho e Tim Salimans. “Classifier-Free Diffusion Guidance”. Em: arXiv preprint arXiv:2207.12598 (2022)
- Robin Rombach et al. “High-Resolution Image Synthesis with Latent Diffusion Models”. Em: CVPR. 2022, pp. 10684–10695
- Jascha Sohl-Dickstein et al. “Deep Unsupervised Learning using Nonequilibrium Thermodynamics”. Em: ICML. 2015, pp. 2256–2265

Referências I

[1] Jonathan Ho, Ajay Jain e Pieter Abbeel. “Denoising Diffusion Probabilistic Models”. Em: NeurIPS. Vol. 33. 2020, pp. 6840–6851.

[2] Jonathan Ho e Tim Salimans. “Classifier-Free Diffusion Guidance”. Em: arXiv preprint arXiv:2207.12598 (2022).

[3] Simon J. D. Prince. Understanding Deep Learning. MIT Press, 2023.

[4] Robin Rombach et al. “High-Resolution Image Synthesis with Latent Diffusion Models”. Em: CVPR. 2022, pp. 10684–10695.

[5] Jascha Sohl-Dickstein et al. “Deep Unsupervised Learning using Nonequilibrium Thermodynamics”. Em: ICML. 2015, pp. 2256–2265.