

TP558 - Tópicos avançados em Machine Learning: *Diffusion Models*



Introdução

- Vocês já ouviram falar ou usaram os modelos de IA: Midjourney, Stable Diffusion, DALL-E?
- Eles são chamados de **modelos de difusão** e geram imagens sintéticas.
 - Eles geram imagens com base em instruções (chamados de *prompts* em inglês).
- Eles também são usados para gerar vídeos, músicas, novas drogas, etc.

Stable Diffusion



DALLE 2

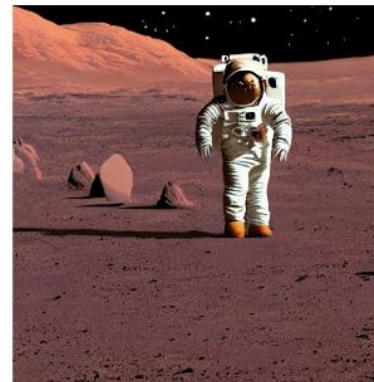


Midjourney



Cherry Blossom near a lake, snowing

Stable Diffusion



DALLE 2



Midjourney



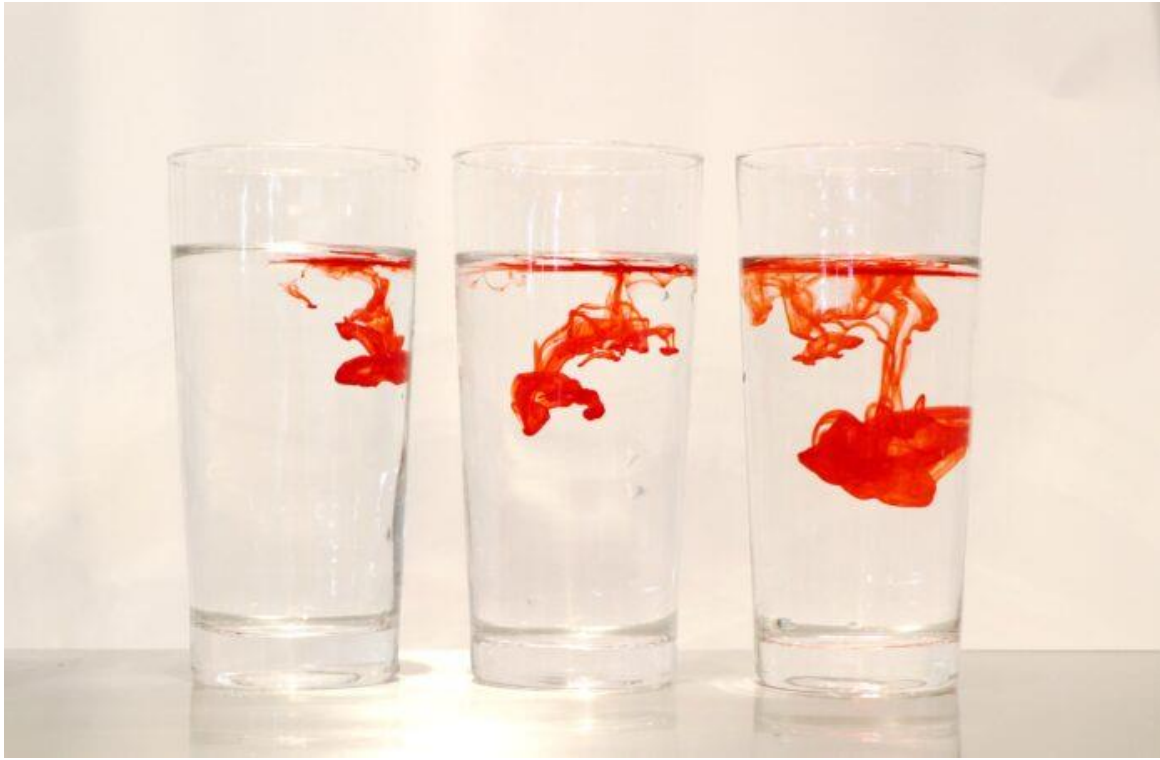
Alone astronaut on Mars, mysterious, colorful, hyper realistic

Introdução



- Eles recebem esse nome devido à sua **semelhança** com o **processo de difusão**, que descreve como partículas ou **moléculas se movem de uma área de alta concentração para uma de baixa concentração** impulsionadas pelo movimento térmico aleatório das partículas.

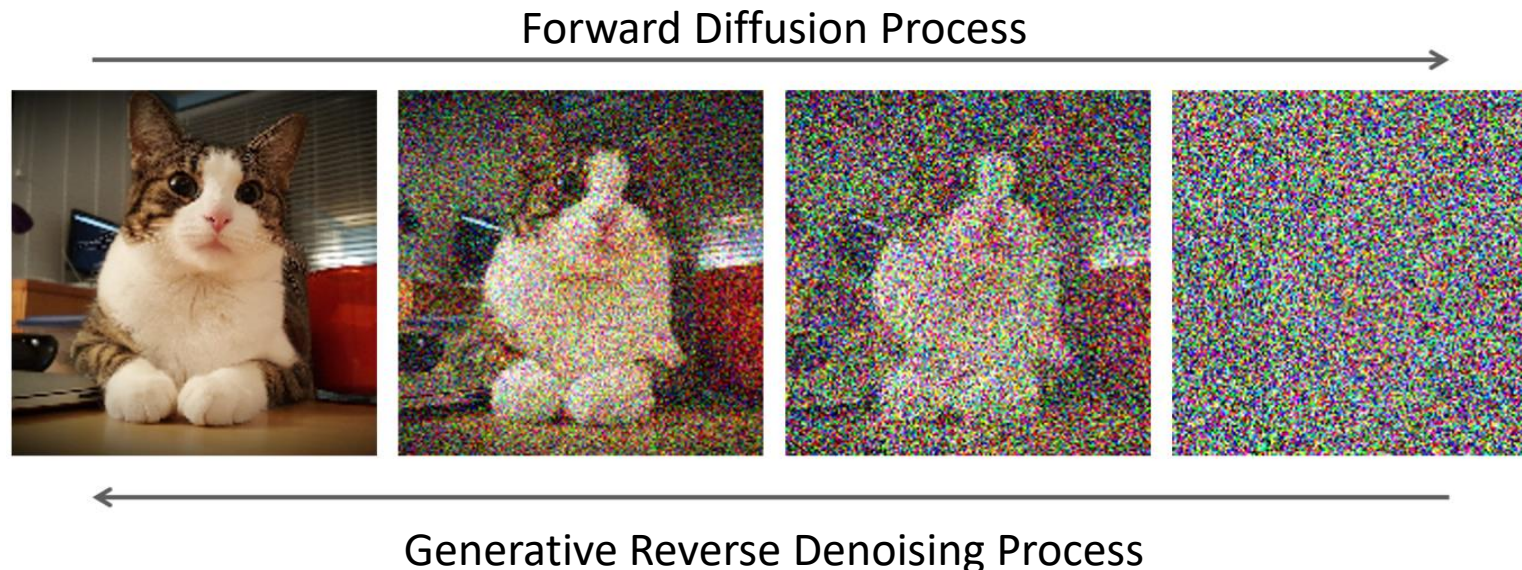
Introdução



- Por exemplo, se adicionarmos uma gota de corante a um copo de água, o corante espalha-se pela água, à medida que as partículas do corante se difundem de uma área de maior concentração (a gota) para uma área de menor concentração (o resto da água).

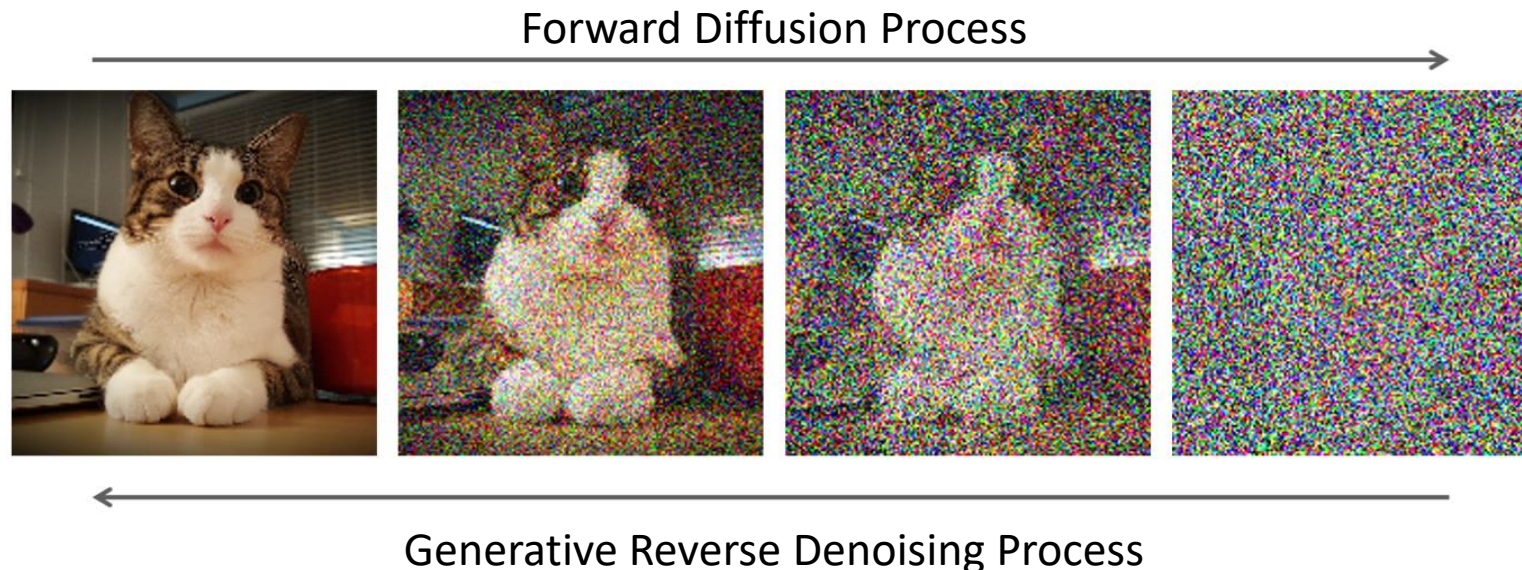
Introdução

- No contexto de aprendizado de máquina, os modelos de difusão geram novos dados ***revertendo um processo de difusão***, ou seja, perda de informação devido à adição de ruído.
- A ideia principal é ***adicionar ruído*** aleatório aos dados e então ***desfazer o processo*** para obter a ***distribuição original*** por trás dos dados ruidosos.



Introdução

- Nesse contexto, a ***difusão transforma passo a passo*** (i.e., iterativamente) ***um sinal estruturado*** (e.g., uma imagem) ***em ruído***.
- Ao simular a difusão, geramos imagens ruidosas a partir das imagens de treinamento e, assim, podemos ***treinar uma rede neural para remover o ruído*** delas.



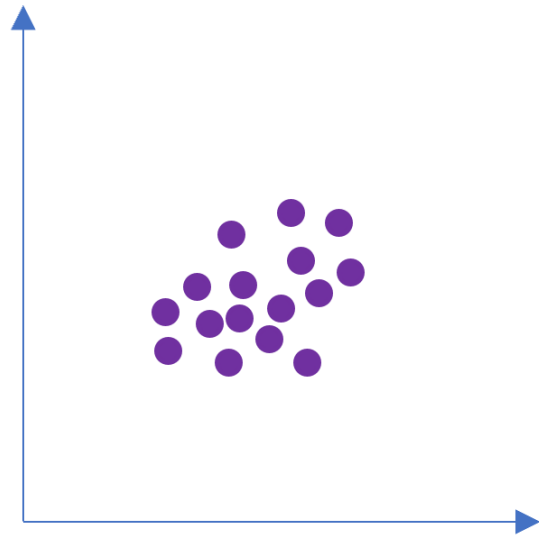
Introdução



- **Resumo:** os modelos de difusão são treinados para *eliminar ruído de imagens ruidosas*.
- Consequentemente, eles aprendem a *gerar novas imagens eliminando iterativamente o ruído* de *amostras puramente ruidosas*.
- Portanto, neste seminário, nós veremos como esses modelos funcionam.

Termodinâmica de não equilíbrio

Distribuição das imagens que ocorrem naturalmente

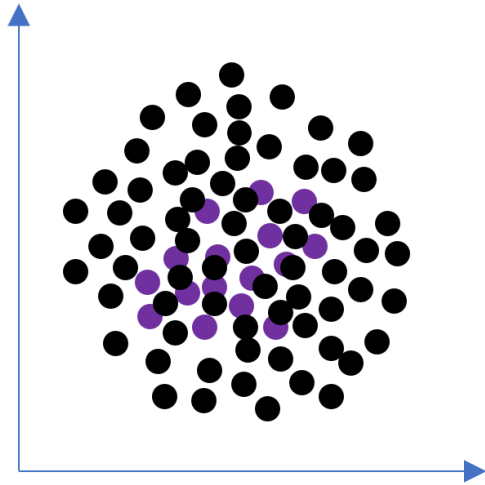


● Imagem no espaço de imagens

- Vamos considerar, por exemplo, a distribuição, q , de todas as imagens que ocorrem naturalmente.
 - A distribuição q é chamada de *distribuição original*, alvo ou objetivo.
- Cada imagem é um ponto no espaço criado por todas as imagens e a distribuição das imagens que ocorrem naturalmente é uma "nuvem" nesse espaço.

Termodinâmica de não equilíbrio

Distribuição das imagens que ocorrem naturalmente + ruído

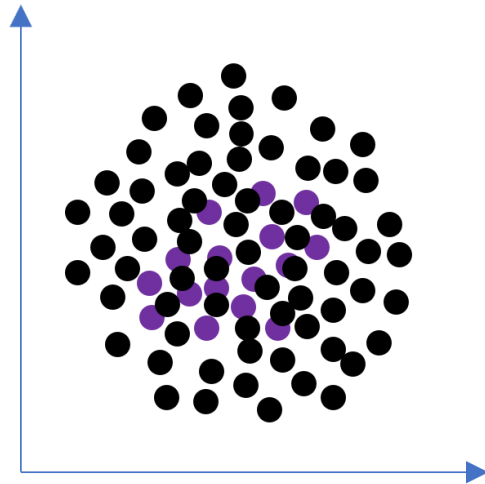


- Imagem no espaço de imagens
- Imagem no espaço de imagens + ruído

- Ao adicionar repetidamente ruído às imagens, a distribuição se *difunde* para o resto do espaço de imagens, até que a nuvem se torne praticamente *indistinguível de uma distribuição Gaussiana*, $N(\mathbf{0}, \mathbf{I})$.
 - N , em geral, é uma distribuição multidimensional, onde $\mathbf{0}$ é o vetor de médias e \mathbf{I} é a matriz identidade.

Termodinâmica de não equilíbrio

Distribuição das imagens que ocorrem naturalmente + ruído

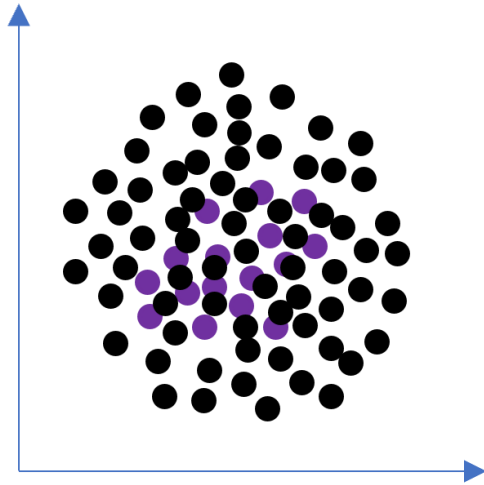


- Imagem no espaço de imagens
- Imagem no espaço de imagens + ruído

- Um modelo que pode *desfazer a difusão* pode então ser usado para extrair amostras da distribuição original, q .
- Ou seja, o modelo *remove o ruído* de uma amostra *até que reste apenas valores retirados de distribuição original*, q .

Termodinâmica de não equilíbrio

Distribuição das imagens que ocorrem naturalmente + ruído

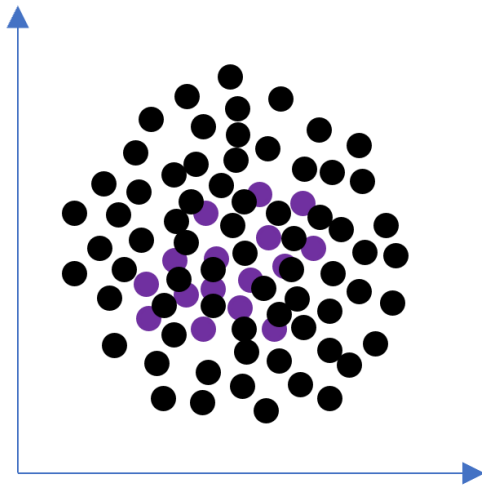


- Imagem no espaço de imagens
- Imagem no espaço de imagens + ruído

- Isto é estudado na **termodinâmica de não equilíbrio**, pois a distribuição inicial, q , **não está em equilíbrio**, ao contrário da distribuição final, N .
- A distribuição de equilíbrio é a distribuição Gaussiana.

Termodinâmica de não equilíbrio

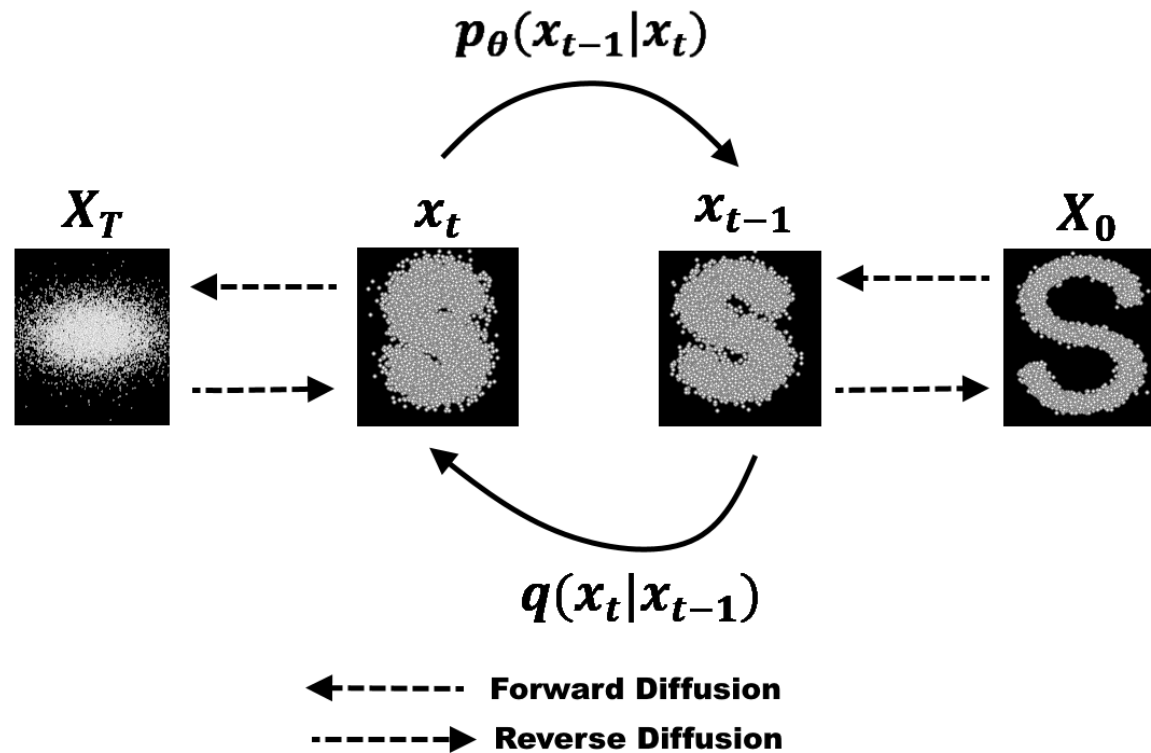
Distribuição das imagens que ocorrem naturalmente + ruído



- Imagem no espaço de imagens
- Imagem no espaço de imagens + ruído

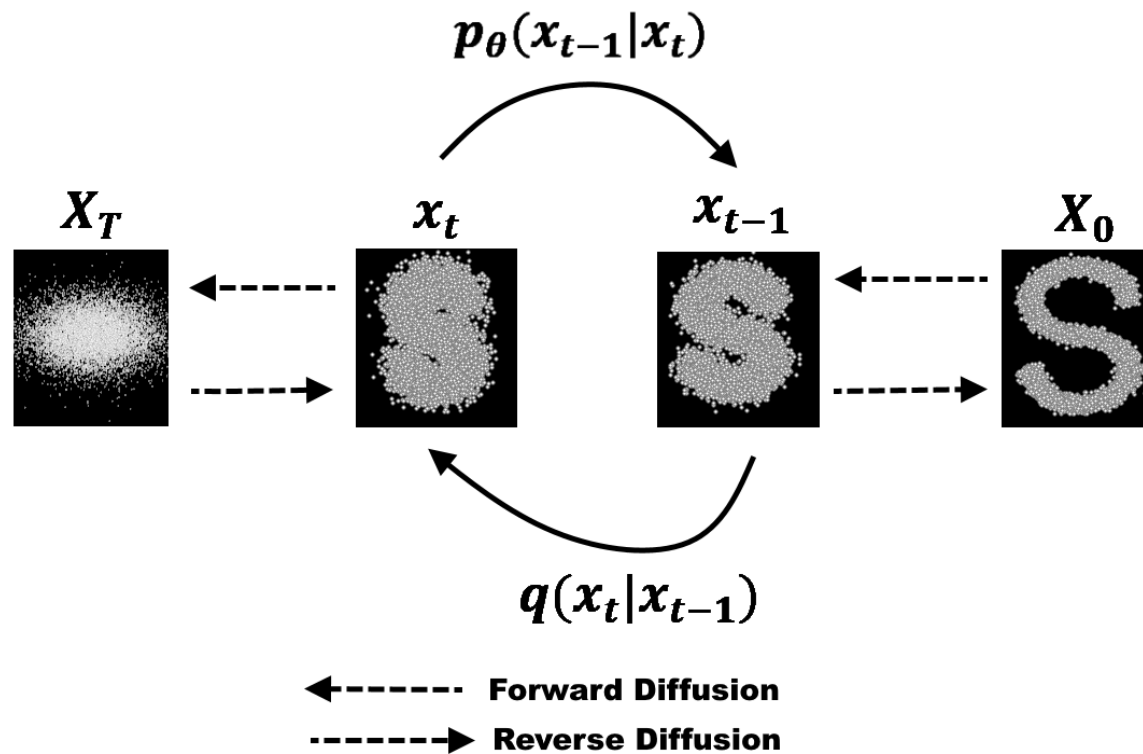
- A distribuição inicial estando fora do equilíbrio, se difunde em direção à distribuição de equilíbrio.
- O objetivo dos modelos de difusão é *aprender um processo reverso de difusão* que gere a *distribuição de probabilidade* de um determinado conjunto de dados *a partir apenas de amostras ruidosas*.

Modelos de difusão



- Um *modelo de difusão* consiste em um *processo direto* (ou difusão), no qual um dado (e.g., uma imagem) tem ruído adicionado a ele progressivamente, e um *processo reverso* (ou difusão reversa), no qual o ruído é transformado novamente em uma amostra da distribuição alvo.

Modelos de difusão



- Na prática, ele é formulado usando uma cadeia de Markov de T passos.
 - Em uma cadeia de Markov, cada **passo (ou estado) depende apenas do anterior.**
- Por essa razão, as probabilidades são condicionais.
- Veremos na sequência como funcionam os processos direto e reverso de difusão.

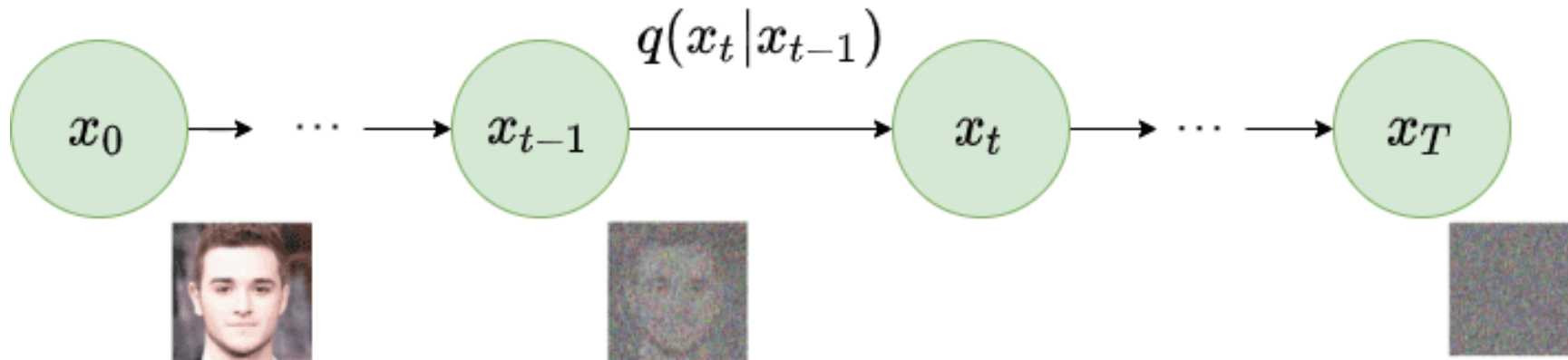
Processo de difusão direta

- Dado uma amostra de dados retirada de uma distribuição de dados real, $\mathbf{x}_0 \sim q(\mathbf{x})$, vamos definir um processo de **difusão direta** no qual adicionamos uma pequena quantidade de ruído gaussiano à amostra em T **passos**, produzindo uma sequência de amostras ruidosas $\mathbf{x}_1, \dots, \mathbf{x}_T$.
 - $q(\mathbf{x})$ é a distribuição de probabilidade a ser aprendida, ou seja, a distribuição alvo.
- Os **tamanhos dos passos** são controlados por um conjunto de variâncias, $\{\beta_t \in [0, 1]\}_{t=1}^T$.
 - β_t pode ser constante ou variar ao longo dos passos até T .
 - Entretanto, resultados mostram que variar β_t ao longo do tempo produz resultados melhores: variação linear, quadrática, cossenoidal, etc. [1].
- A variação do valor de β_t garante que \mathbf{x}_T **tenha praticamente uma distribuição Gaussiana isotrópica** para T suficientemente grande.

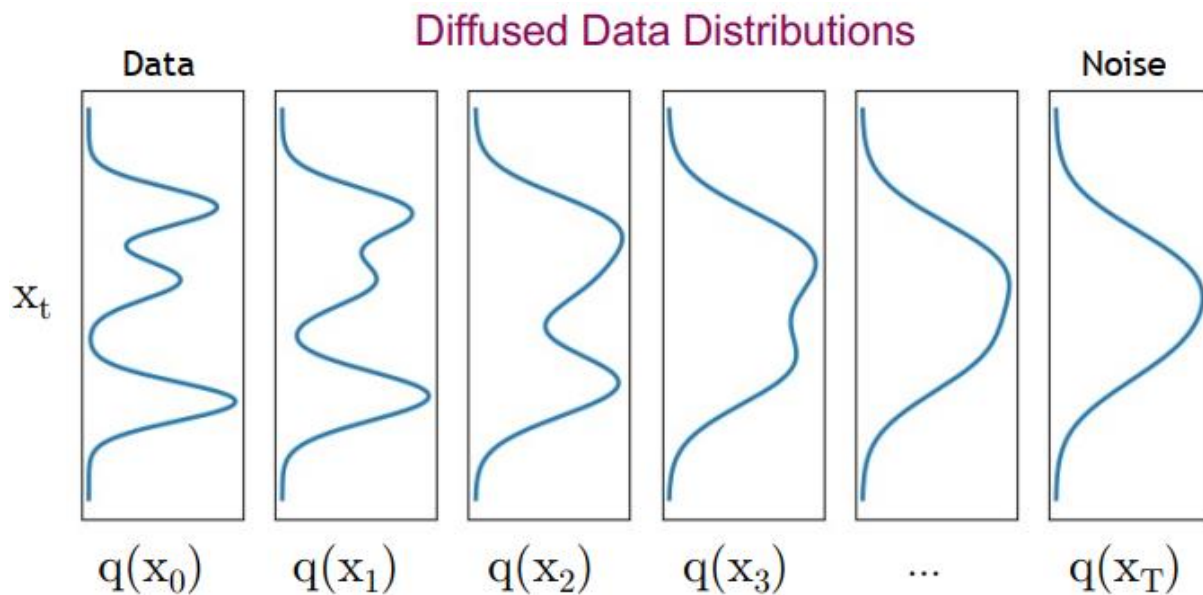
Processo de difusão direta

- Em cada passo, t , da cadeia de Markov adicionamos ruído gaussiano com variância β_t à \mathbf{x}_{t-1} , gerando uma nova variável \mathbf{x}_t com **distribuição condicional**

$q(\mathbf{x}_t | \mathbf{x}_{t-1}) = \sqrt{1 - \beta_t} \mathbf{x}_{t-1} + \sqrt{\beta_t} \boldsymbol{\epsilon} = N(\mathbf{x}_t; \boldsymbol{\mu}_t = \sqrt{1 - \beta_t} \mathbf{x}_{t-1}, \boldsymbol{\Sigma}_t = \beta_t \mathbf{I})$,
onde $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \mathbf{I})$.



Processo de difusão direta



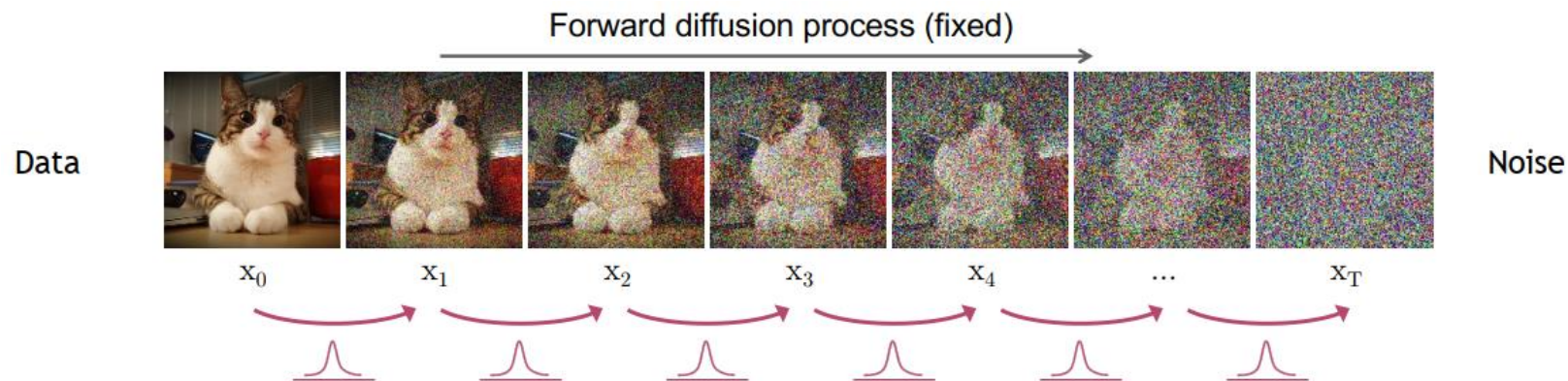
- A amostra de dados, x_0 , perde gradualmente suas características distinguíveis à medida que o passo se torna maior.
- Eventualmente, quando $T \rightarrow \infty$, x_T é equivalente a uma distribuição Gaussiana isotrópica.

Processo de difusão direta

- Notem que $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ continua sendo uma distribuição normal.
- Percebam também que $\mathbf{x}_1, \dots, \mathbf{x}_T$ têm a *mesma dimensão* de \mathbf{x}_0 .
- Assim, podemos ir a partir do dado de entrada \mathbf{x}_0 até \mathbf{x}_T , da seguinte forma

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

- Esse processo que vai de \mathbf{x}_0 até \mathbf{x}_T é chamado de *trajetória (direta)*.



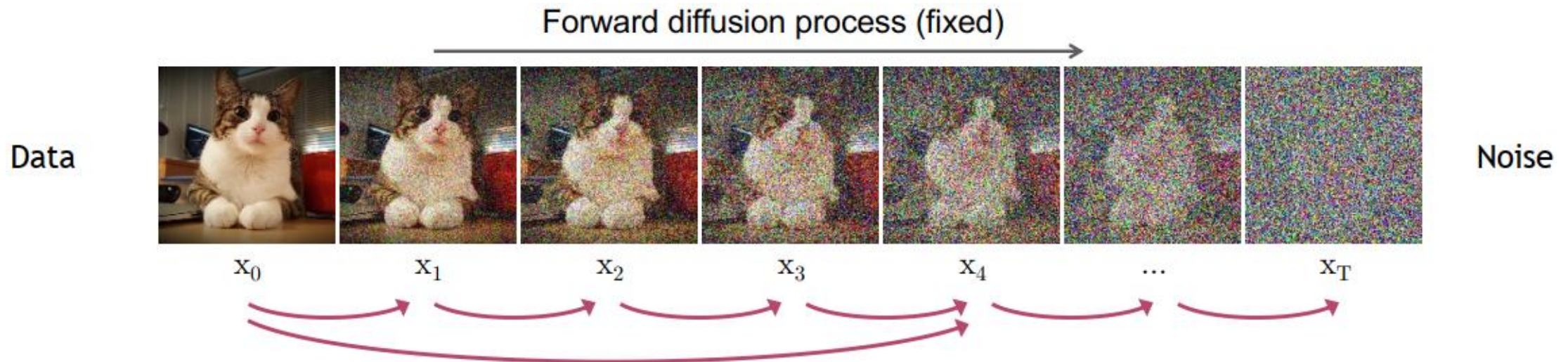
Processo de difusão direta

- Usando o **truque da reparametrização**, podemos amostrar \mathbf{x}_t em **qualquer instante de tempo**, t , como

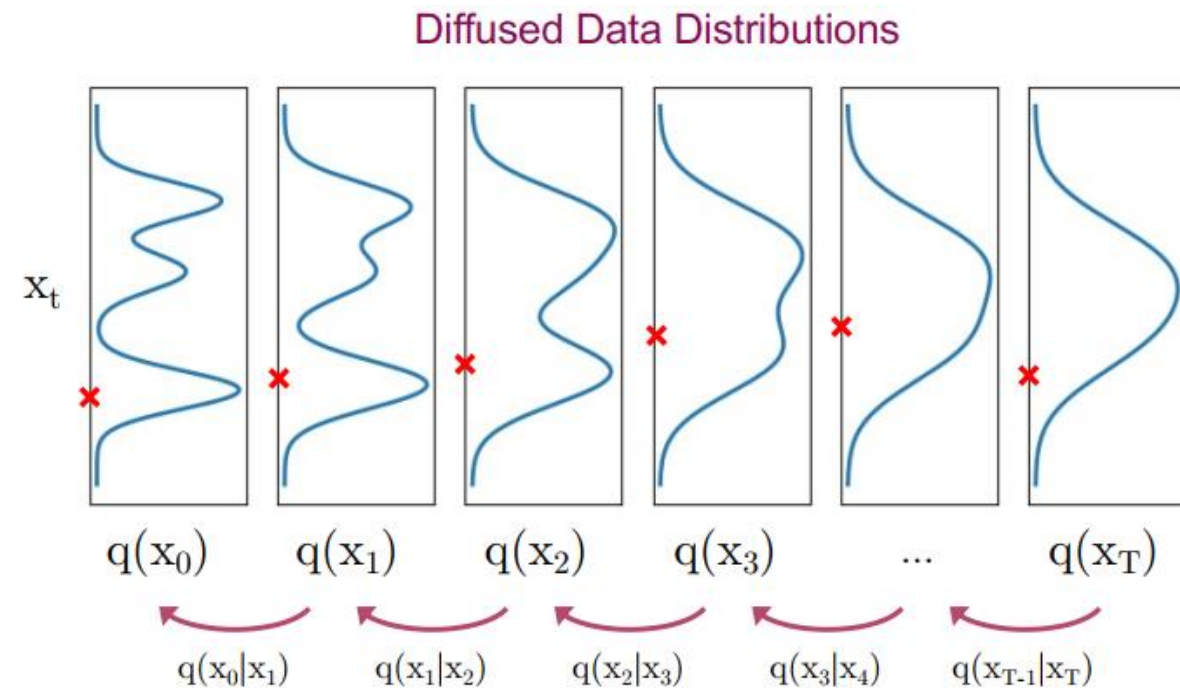
$$q(\mathbf{x}_t | \mathbf{x}_0) = N(\mathbf{x}_t; \sqrt{\bar{\alpha}_t} \mathbf{x}_0, (1 - \bar{\alpha}_t) \mathbf{I}),$$

onde $\alpha_t = 1 - \beta_t$ e $\bar{\alpha}_t = \prod_{i=1}^t \alpha_i$. Assim

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}.$$



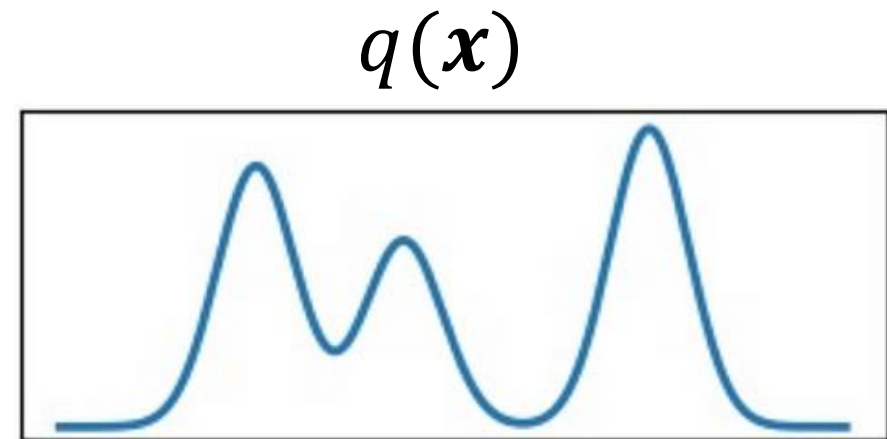
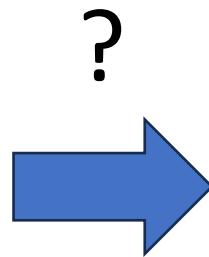
Processo de difusão reversa



- A “mágica” dos modelos de difusão ocorre no processo inverso.
- Se pudermos reverter o processo direto e amostrar da distribuição $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$, poderemos recriar a amostra verdadeira, \mathbf{x}_0 , a partir de uma amostra de ruído Gaussiano (normal padrão), $\mathbf{x}_T \sim N(\mathbf{0}, \mathbf{I})$.
 - $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ é chamada de *distribuição de remoção do ruído*.

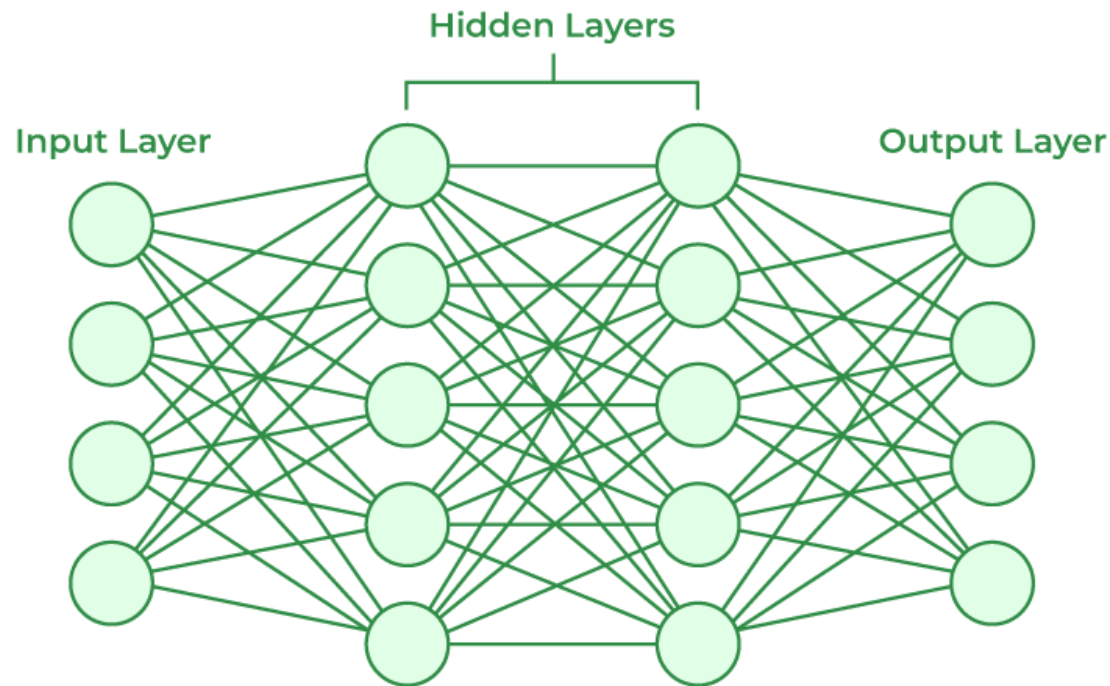
Processo de difusão reversa

- Na prática, nós não conhecemos a distribuição condicional $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$.
- É algo intratável, dado que estimativas estatísticas de $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ requerem cálculos envolvendo a *distribuição real dos dados*, o que *não existe ou é desconhecida*.



Então como podemos modelar o processo de difusão reversa?

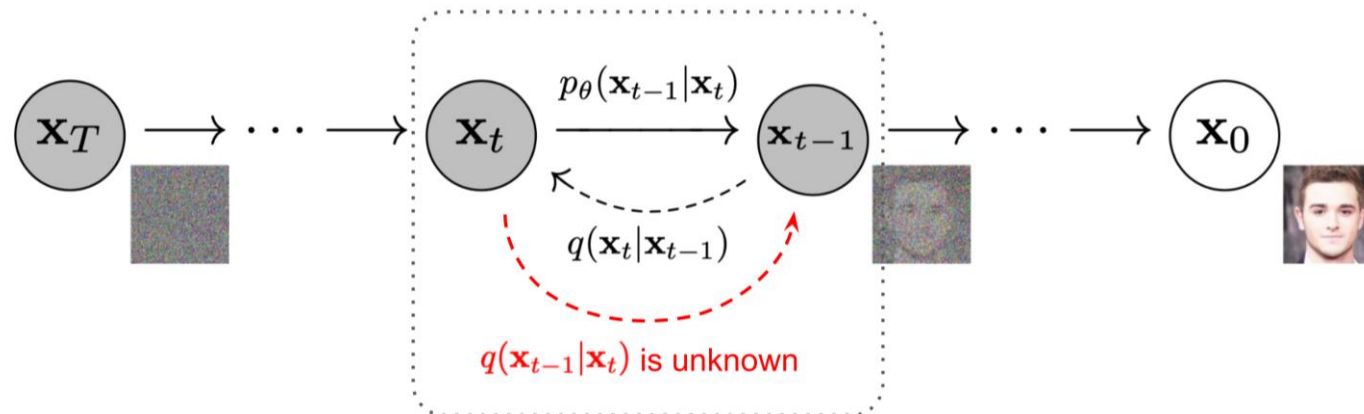
Processo de difusão reversa



- Nós **aproximamos** o **processo reverso**, $q(x_{t-1}|x_t)$, com um modelo parametrizável, p_θ , (e.g., uma **rede neural**), onde θ são os pesos da rede neural, atualizados pelo **algoritmo do gradiente descendente**.

Processo de difusão reversa

- A distribuição $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ pode ser *assumida* como sendo Gaussiana *se β_t for pequeno o suficiente em cada passo, t , da difusão direta*.
- Assim, escolhemos p_θ como sendo uma *distribuição Gaussiana* e *parametrizamos sua média e matriz de covariância*
$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = N(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)).$$
- **OBS.:** A média e a variância também estão *condicionadas ao passo, t* , o qual dita o nível de ruído.



O que a rede deve prever?

Processo de difusão reversa

- A rede recebe dois argumentos, \mathbf{x}_t e t , e gera em sua saída um vetor $\boldsymbol{\mu}_\theta(\mathbf{x}_t, t)$ e uma matriz $\boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)$, de modo que cada passo do processo de difusão direta possa ser aproximadamente desfeito por

$$\mathbf{x}_{t-1} \sim N(\boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)).$$

- Começando com ruído puro, $p(\mathbf{x}_T) = N(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$, o modelo aprende a **distribuição conjunta do processo reverso** de T a 0 como

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

- A distribuição $p_\theta(\mathbf{x}_{0:T})$ também é chamada de **trajetória (reversa)**.

Processo de difusão reversa

- Os autores do artigo DDPM [1], fazem uma simplificação, onde $\Sigma_{\theta}(\mathbf{x}_t, t) = \sigma_t^2 \mathbf{I}$, onde $\sigma_t^2 = \beta_t$ ou $\sigma_t^2 = \tilde{\beta}_t$, onde $\tilde{\beta}_t = \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t} \beta_t$.
 - Ambos valores de σ_t^2 deram resultados semelhantes .
- Desta forma, a *rede neural precisa aprender apenas a média* $\mu_{\theta}(\mathbf{x}_t, t)$ da distribuição condicional de probabilidade,

$$p_{\theta}(\mathbf{x}_{t-1} | \mathbf{x}_t) = N \left(\mathbf{x}_{t-1}; \underbrace{\mu_{\theta}(\mathbf{x}_t, t)}_{\text{rede neural}}, \sigma_t^2 \mathbf{I} \right).$$

- Agora veremos como o modelo é treinado.

Treinando um modelo de difusão

- Como qualquer outro modelo de aprendizado supervisionado, precisamos definir uma função de erro (ou de perda).
- O treinamento é realizado minimizando-se o *limite inferior variacional*, o qual é dado por

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

- Que nada mais é do que a função de perda para o treinamento do modelo.

Treinando um modelo de difusão



Treinando um modelo de difusão

- Os autores de [1] mostram que L pode ser simplificado como uma **soma de perdas para os $T + 1$ intervalos de tempo**

$$\begin{aligned} L &= \mathbb{E}_q \left[\underbrace{D_{KL}(q(\mathbf{x}_T | \mathbf{x}_0) || p(\mathbf{x}_T))}_{L_T} \right. \\ &\quad \left. + \sum_{t>1} \underbrace{D_{KL}(q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) || p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t))}_{L_{t-1}} \underbrace{- \log p_\theta(\mathbf{x}_0 | \mathbf{x}_1)}_{L_0} \right] \\ &= L_0 + L_1 + \dots + L_{t-1} + L_T, \end{aligned}$$

onde D_{KL} é a medida de divergência de **Kullback-Leibler** (KL) e $q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0)$ é uma **distribuição posterior tratável**, por estar também condicionada à \mathbf{x}_0 [1].

Treinando um modelo de difusão

- A **distribuição posterior da etapa direta** é dada por

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

onde

$$\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{\sqrt{\bar{\alpha}_{t-1}\beta_t}}{1 - \bar{\alpha}_t} \mathbf{x}_0 + \frac{\sqrt{1 - \beta_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} \mathbf{x}_t.$$

- D_{KL} é uma **medida de distância estatística** de quanto uma distribuição de probabilidade P difere de uma distribuição de referência Q

$$D_{KL}(P||Q) = \int_{-\infty}^{+\infty} p(x) \log \left(\frac{p(x)}{q(x)} \right) dx.$$

- Uma divergência de KL igual a 0 indica que as distribuições P e Q são muito parecidas, até mesmo iguais.

Treinando um modelo de difusão

- Como $\beta_1, \beta_2, \dots, \beta_T$ são valores constantes e conhecidos, o termo L_T **não tem nenhum parâmetro a ser aprendido** e, portanto, ele é ignorado durante o treinamento.
- O termo $p_\theta(\mathbf{x}_0|\mathbf{x}_1)$ de $L_0 = -\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)$ é calculado como

$$p_\theta(\mathbf{x}_0|\mathbf{x}_1) = \prod_{i=1}^D \int_{\delta_-(x_0^i)}^{\delta_+(x_0^i)} N(x; \mu_\theta^i(\mathbf{x}_1, 1), \sigma_1^2) dx,$$

onde D é a dimensionalidade dos dados, i é um sobrescrito que indica um elemento.

Treinando um modelo de difusão

- Adicionalmente, $\delta_-(x_0^i)$ e $\delta_+(x_0^i)$ são definidos como

$$\delta_-(x) = \begin{cases} -\infty & \text{se } x = -1 \\ x - \frac{1}{255} & \text{se } x > -1 \end{cases},$$
$$\delta_+(x) = \begin{cases} \infty & \text{se } x = 1 \\ x + \frac{1}{255} & \text{se } x < 1 \end{cases}.$$

- As imagens consistem de valores inteiros $\{0, \dots, 255\}$ normalizados para o intervalo -1 a 1, garantindo que o processo reverso opere com entradas escalonadas, começando de $p(\mathbf{x}_T)$.
- Entretanto, os autores de [1] obtiveram melhores resultados ao ignorar esse termo.

Treinando um modelo de difusão

- Dado que $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ e $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ são distribuições Gaussianas, a divergência de KL assume uma forma simples e fechada dada por

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

onde C é uma constante que não depende de θ e, portanto, pode ser ignorada.

- Percebam que minimizar L_{t-1} significa *minimizar a diferença (ou distância) entre as médias de duas distribuições Gaussianas*.
- Portanto, a parametrização mais direta de $\boldsymbol{\mu}_\theta$ é *um modelo que prevê $\tilde{\boldsymbol{\mu}}_t$, i.e., a média posterior do processo de difusão*.

Treinando um modelo de difusão

- Substituindo

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}$$

na definição de $\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0)$, podemos reescrever a função L_{t-1} como

$$L_{t-1} = \mathbb{E}_q \left[\frac{1}{2\sigma_t^2} \left\| \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \boldsymbol{\epsilon} \right) - \boldsymbol{\mu}_\theta(\mathbf{x}_t, t) \right\|^2 \right].$$

- A equação acima nos mostra que $\boldsymbol{\mu}_\theta$ (i.e., o modelo) deve prever

$$\frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \boldsymbol{\epsilon} \right)$$

dado \mathbf{x}_t e t .

Treinando um modelo de difusão

- Dado que \mathbf{x}_t e t estão disponíveis como entrada do modelo, podemos fazer a seguinte parametrização

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

onde $\boldsymbol{\epsilon}_\theta$ é uma função aproximadora usada para prever $\boldsymbol{\epsilon}$ a partir de \mathbf{x}_t .

- Assim, usando a parametrização anterior, L_{t-1} pode ser reescrita como

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

Treinando um modelo de difusão

$$L_{t-1} = \mathbb{E}_q \left[\lambda \left\| \epsilon - \epsilon_{\theta} \left(\underbrace{\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon}_{\mathbf{x}_t}, t \right) \right\|^2 \right].$$

- Portanto, o modelo é treinado para **prever o ruído adicionado à amostra \mathbf{x}_t** .
- Notem que a equação acima é o **erro quadrático médio** entre o ruído adicionado no **processo direto** e o ruído **predito pelo modelo**.
- Portanto, a rede neural é otimizada usando-se o **erro quadrático médio** entre o ruído Gaussiano verdadeiro e o predito.
- **OBS.:** Os autores de [1] relataram que a qualidade das imagens geradas é melhor se o termo λ for descartado.

O algoritmo de treinamento

Algorithm 1 Training

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

Algoritmo retirado de [1].

- Retiramos uma amostra aleatória, \mathbf{x}_0 (e.g., imagem), da distribuição de dados real desconhecida $q(\mathbf{x}_0)$.
- Amostramos um nível de ruído, t , uniformemente distribuído entre 1 e T .
- Amostramos ruído Gaussiano normal padrão e corrompemos \mathbf{x}_0 com o nível de ruído dado por t , i.e.,
 - $\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon$.
- A rede neural é treinada para prever o ruído adicionado à amostra \mathbf{x}_t .
 - Ou seja, prever o ruído aplicado à \mathbf{x}_0 com base na sequência β_t .

O algoritmo de amostragem (i.e., geração)

Após o treinamento, para retirar uma amostra da distribuição original, segue-se o algoritmo abaixo.

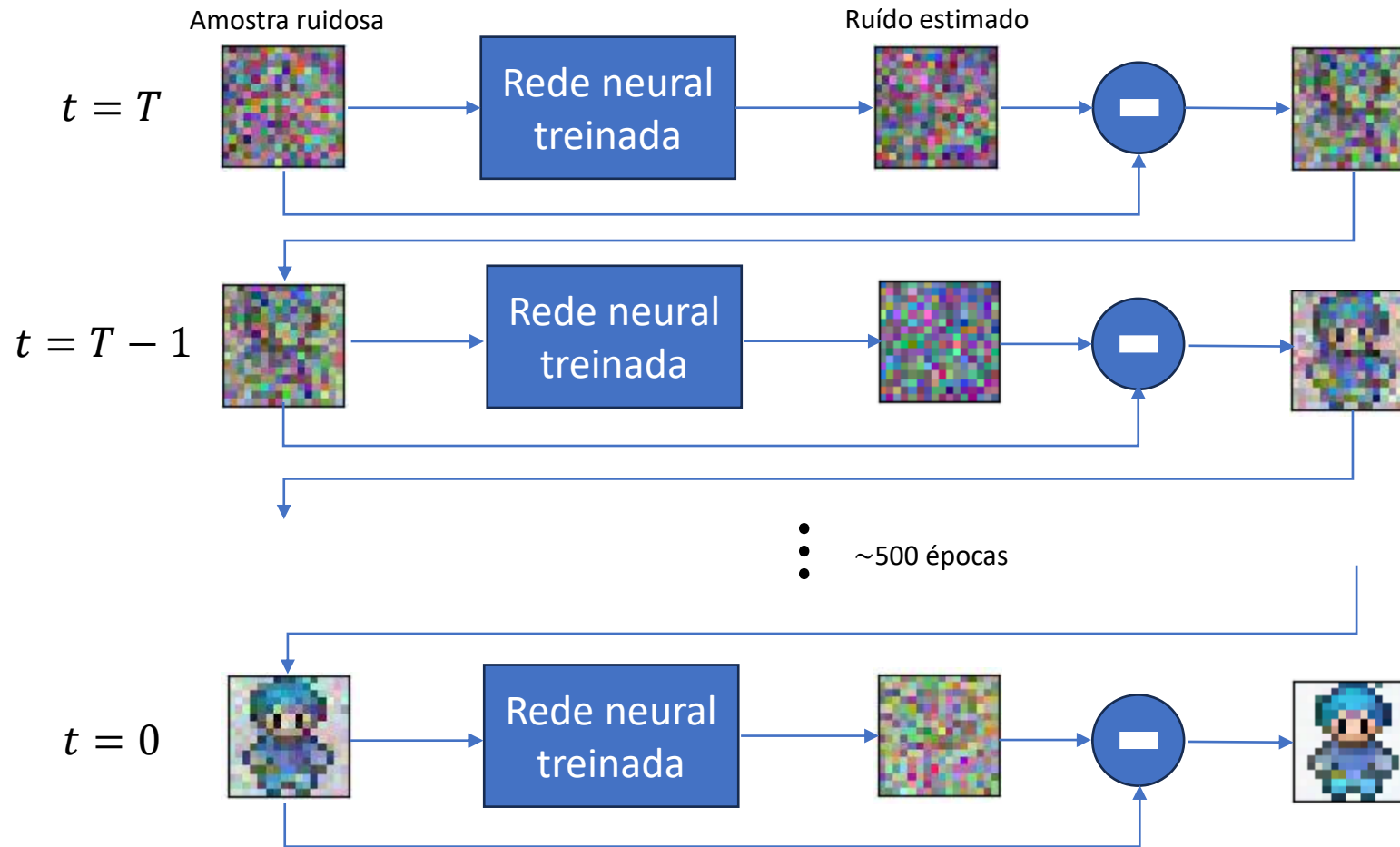
Algorithm 2 Sampling

```
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:    $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left( \mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_{\theta}(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$ 
5: end for
6: return  $\mathbf{x}_0$ 
```

Algoritmo retirado de [1].

- Retira-se uma amostra puramente Gaussiana, \mathbf{x}_T .
- Retira-se uma amostra puramente Gaussiana, \mathbf{z} .
- A rede neural estima o ruído adicionado a \mathbf{x}_t e o remove.
- Antes da nova iteração, ruído Gaussiano, \mathbf{z} , é adicionado à \mathbf{x}_{t-1} .
- Esse processo é repetido até que $t = 1$,
 - Quando $t = 1$, não se adiciona ruído a \mathbf{x}_t .
- Ao final, o algoritmo retorna a amostra da distribuição original, \mathbf{x}_0 .

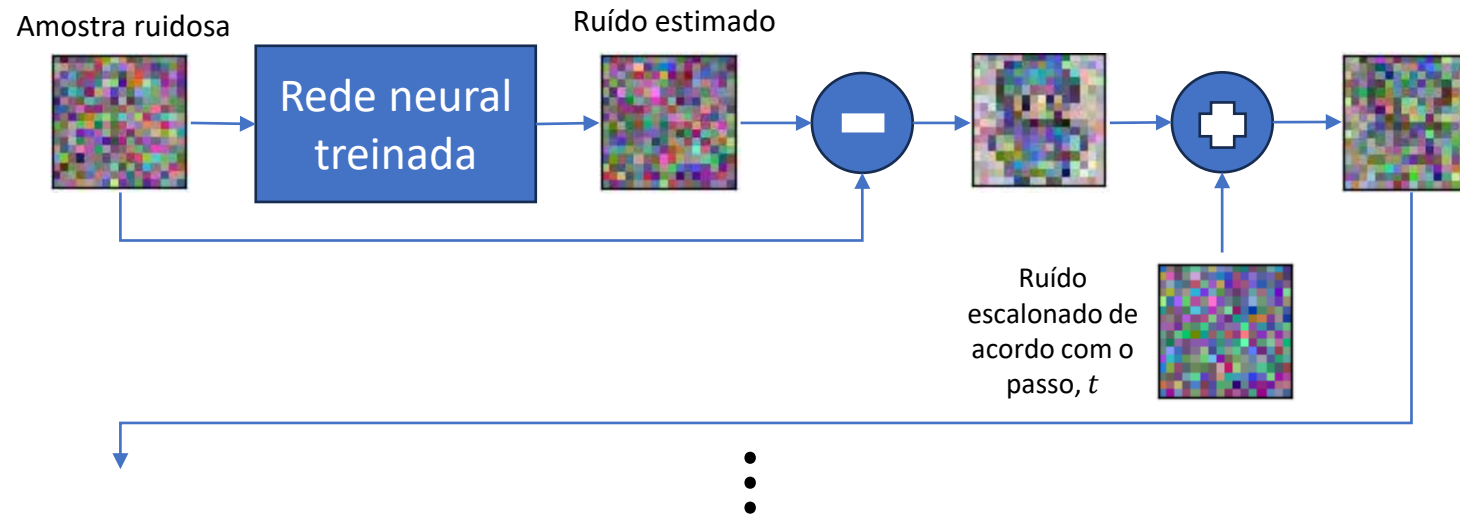
O algoritmo de amostragem (i.e., geração)



OBS.: Exemplo de um modelo de difusão treinado com um conjunto de imagens de *sprites*. Um *sprite* é uma imagem de duas dimensões que pode representar um personagem de um jogo, por exemplo

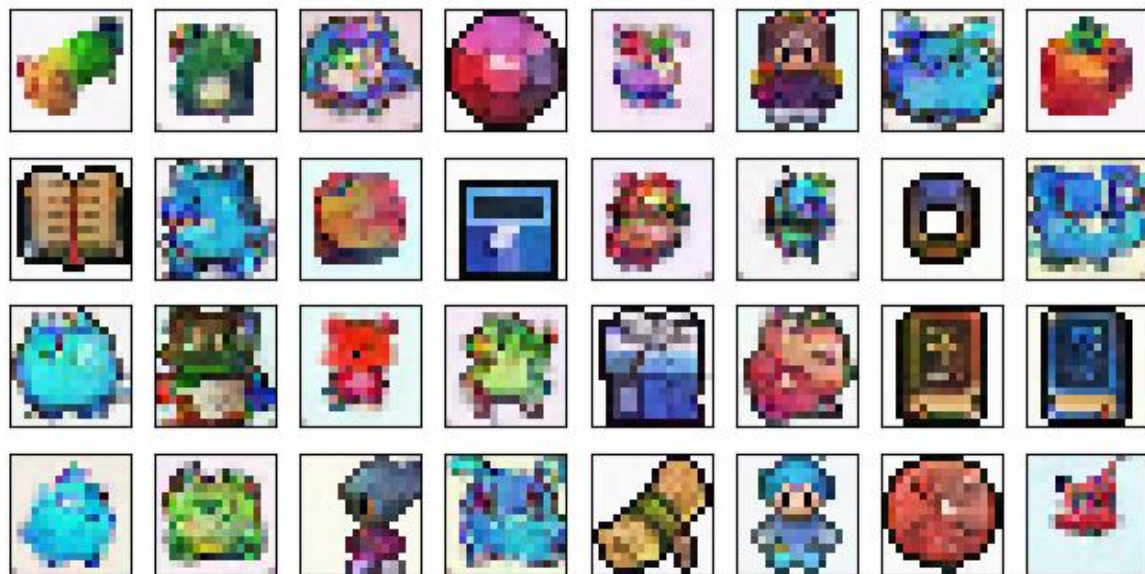
- Em cada passo, a rede prediz o ruído adicionado, até que tenhamos uma imagem da distribuição alvo.

Detalhes do algoritmo de amostragem

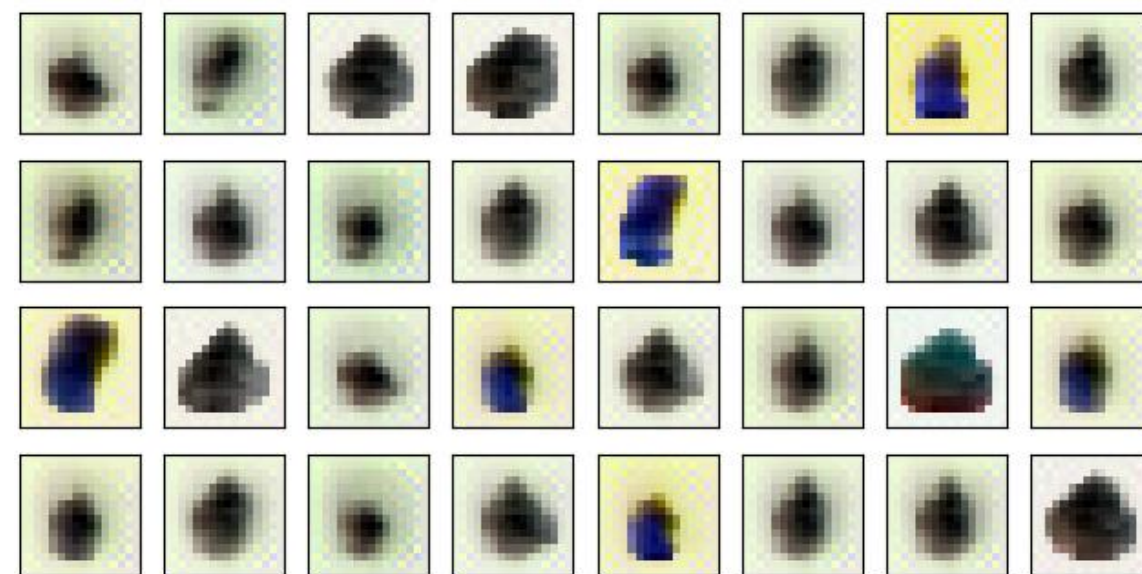


- A rede neural espera ruído Gaussiano normal como entrada.
- Porém, quando removemos o ruído, a amostra deixa de ser distribuída desta forma.
- Portanto, o que devemos fazer após cada passo e antes do próximo é adicionar ruído com variância baseada no passo atual, t .

Detalhes do algoritmo de amostragem



Com adição de ruído antes do próximo passo.



Sem adição de ruído antes do próximo passo.

- Empiricamente, a adição de ruído ajuda a estabilizar a rede neural para que ela não entre em *colapso e gere amostras próximas da média do conjunto de dados*.

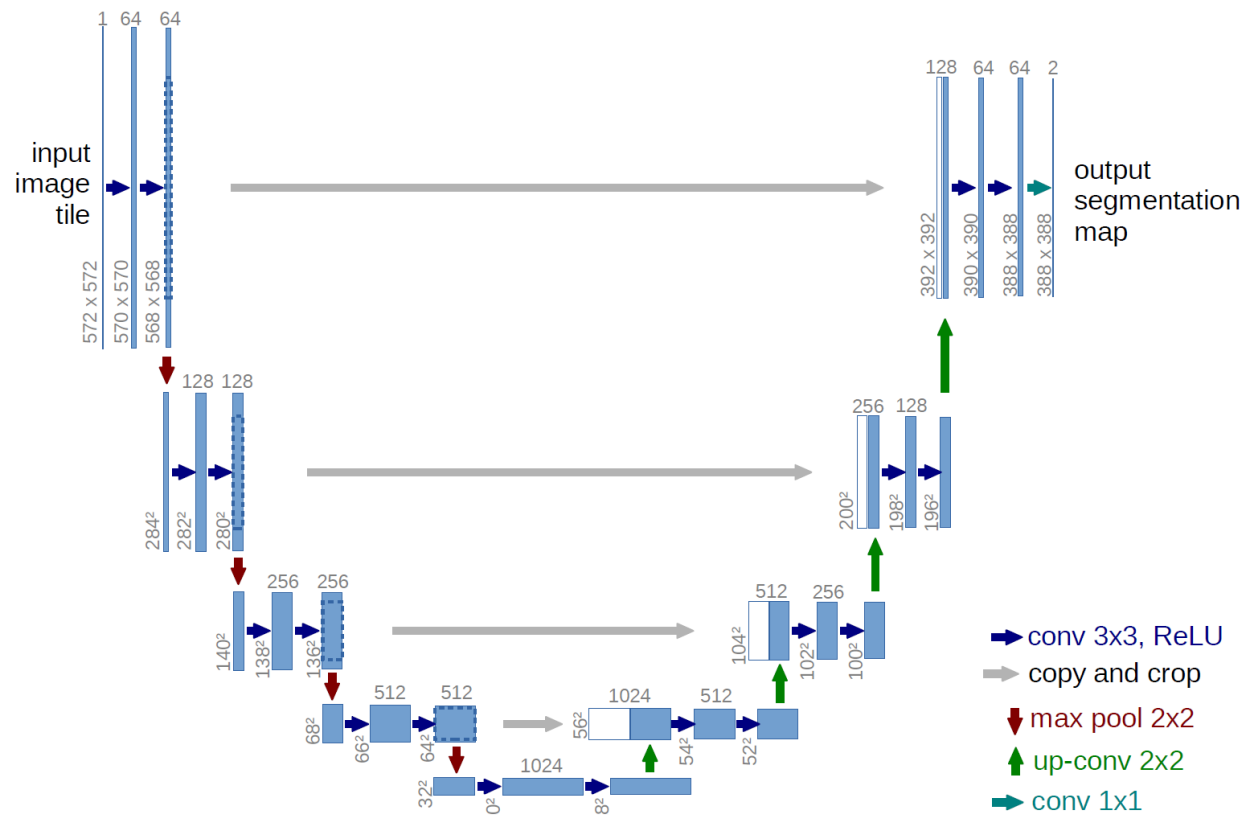
OK, aprendemos os processos direto e reverso de difusão, mas e a rede neural usada, qual é a sua arquitetura, suas características, etc.?

U-Net

$$\epsilon_{\theta}(\underbrace{\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon}_{x_t}, t)$$

- Como vimos, a rede neural recebe uma amostra (e.g., imagem), x_t , com ruído em um determinado passo, t , e retorna uma predição do ruído adicionado a x_t .
- Lembrem-se que o *ruído predito deve ter a mesma dimensão da amostra de entrada*.

U-Net



- Assim, os autores de [1] optaram por usar uma U-Net, introduzida por Ronneberger et al. em 2015 para *segmentação de imagens médicas* [2].
- U-Net é uma rede neural convolucional.
- Ela é chamada de U-Net por ser praticamente simétrica.

U-Net



(a) Image



(b) Semantic segmentation



(c) Instance segmentation



(d) Panoptic segmentation

- A segmentação de imagens é uma técnica de visão computacional que *particiona uma imagem em grupos de pixels* (i.e., segmentos de imagem) para detecção de objetos.
 - Ou seja, *classifica os pixels em classes*.
- Existem três tipos de segmentação.

U-Net



(a) Image



(b) Semantic segmentation



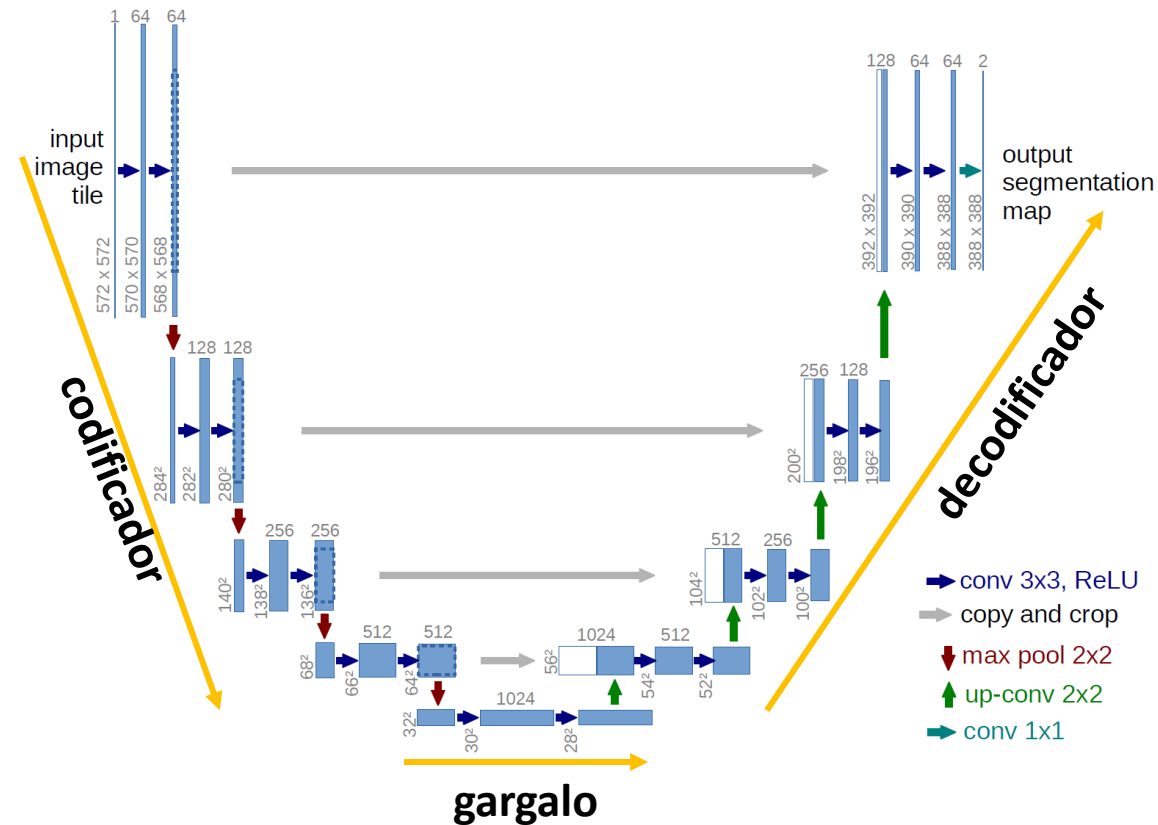
(c) Instance segmentation



(d) Panoptic segmentation

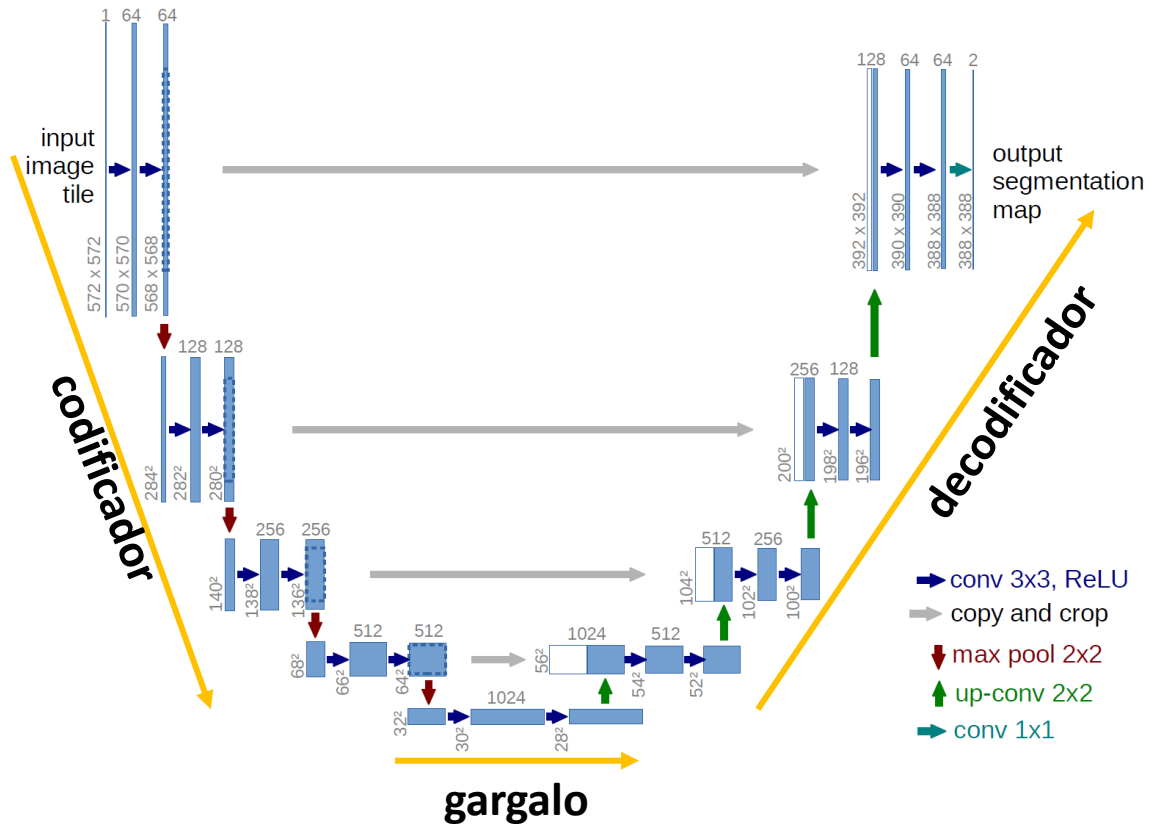
- **Segmentação semântica:** detecta pixels da mesma *classe* (e.g., pessoas).
- **Segmentação por instância:** detecta pixels de *diferentes instâncias de uma classe* (e.g., diferentes pessoas).
- **Segmentação panóptica:** combina as duas anteriores.
- No caso dos modelos de difusão, a U-Net é usada para prever ruído.

U-Net



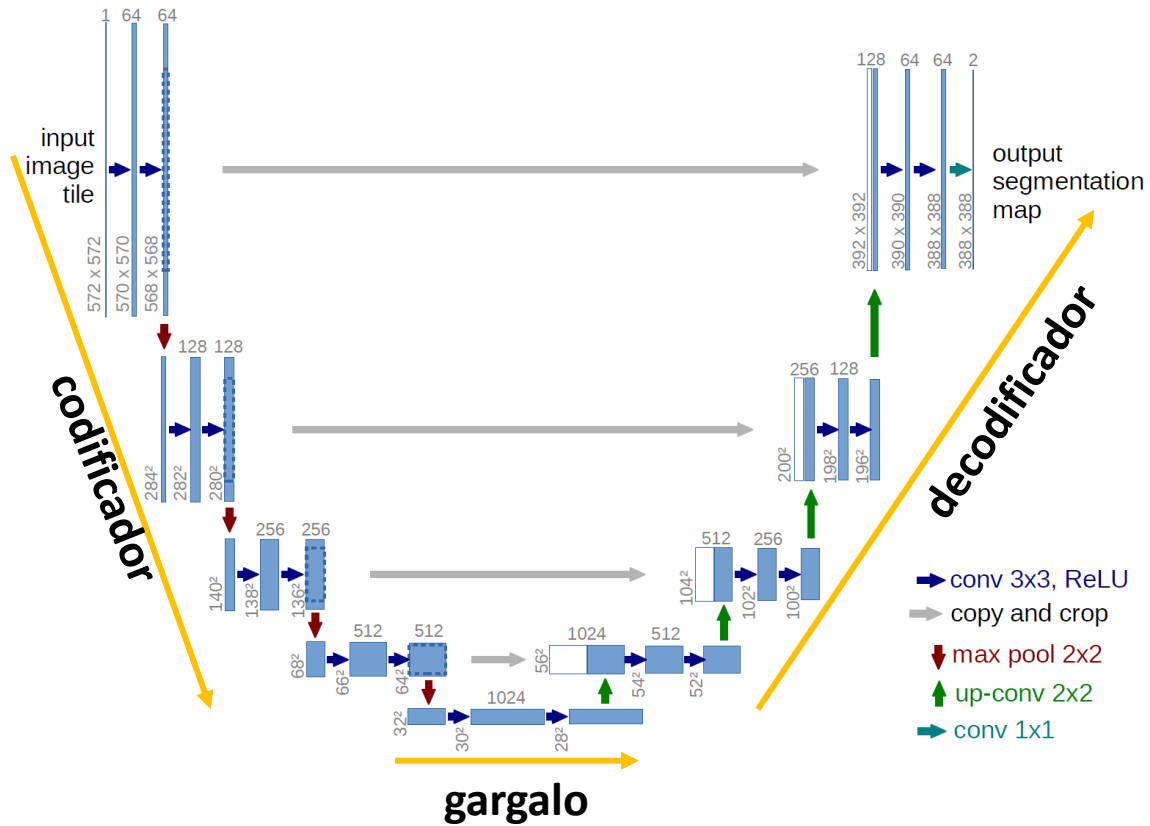
- Uma U-Net é composta por 3 partes: codificador, gargalo e decodificador.

U-Net



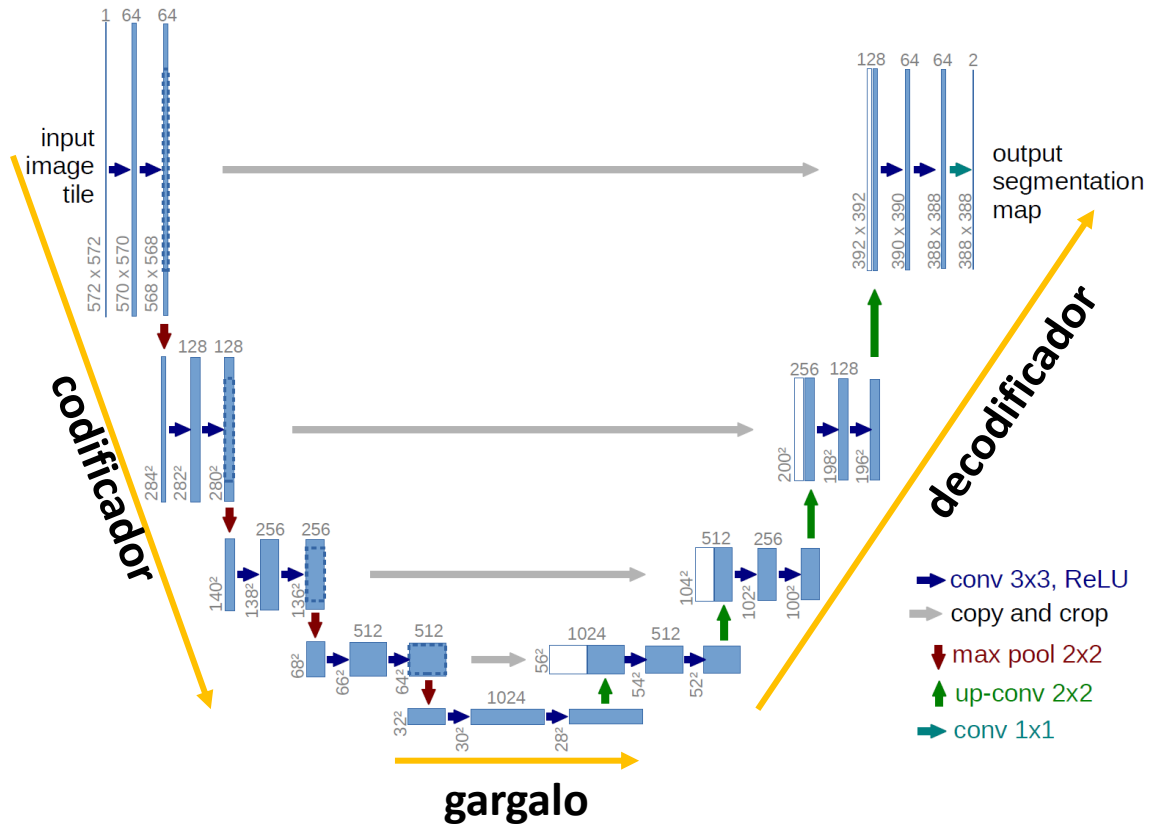
- O codificador transforma uma imagem em uma representação oculta de menor resolução.
- Ou seja, ele torna a entrada *menor* em termos de *resolução espacial*.

U-Net



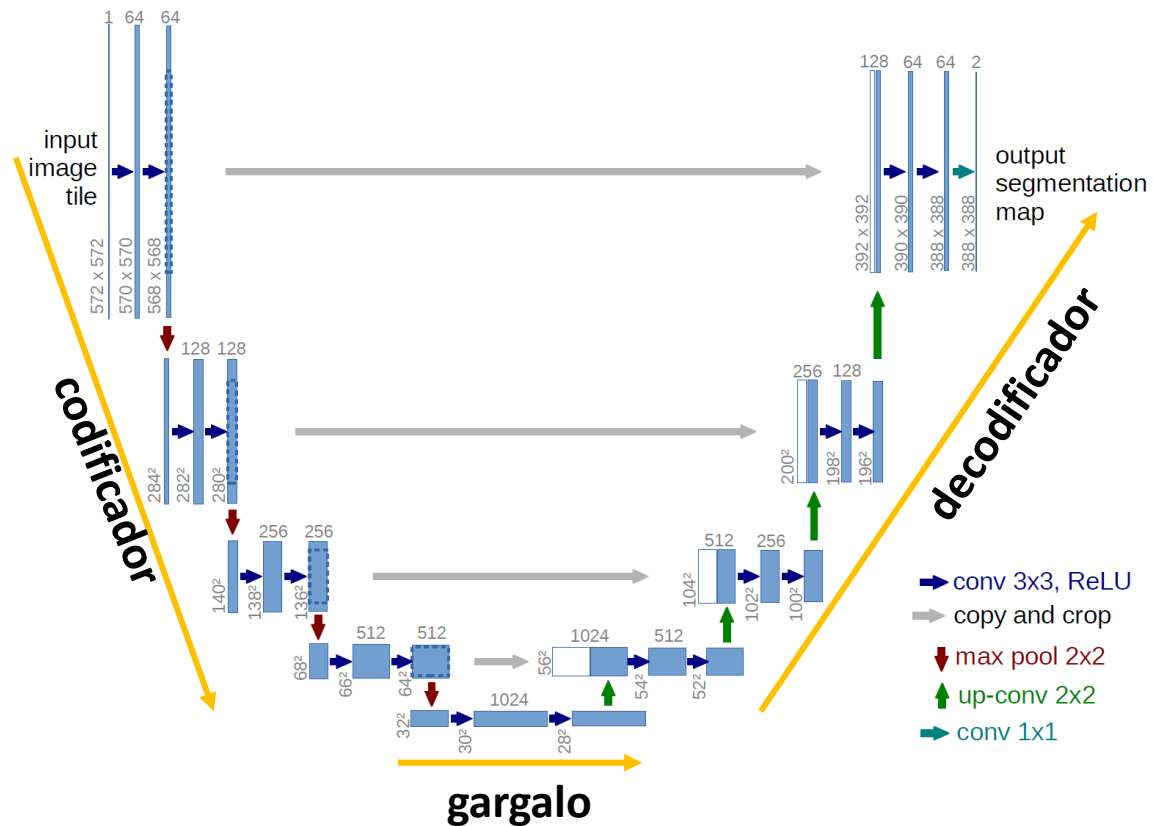
- O “*gargalo*” fica no meio da arquitetura, entre o codificador e o decodificador.
- Ele força o modelo a *aprender uma forma comprimida* da entrada.
 - O gargalo encontra *informações latentes*, i.e., ocultas, nos dados de entrada.
- Ele garante que a rede *aprenda apenas as informações mais importantes*, as quais serão usadas para reconstruir a entrada.

U-Net



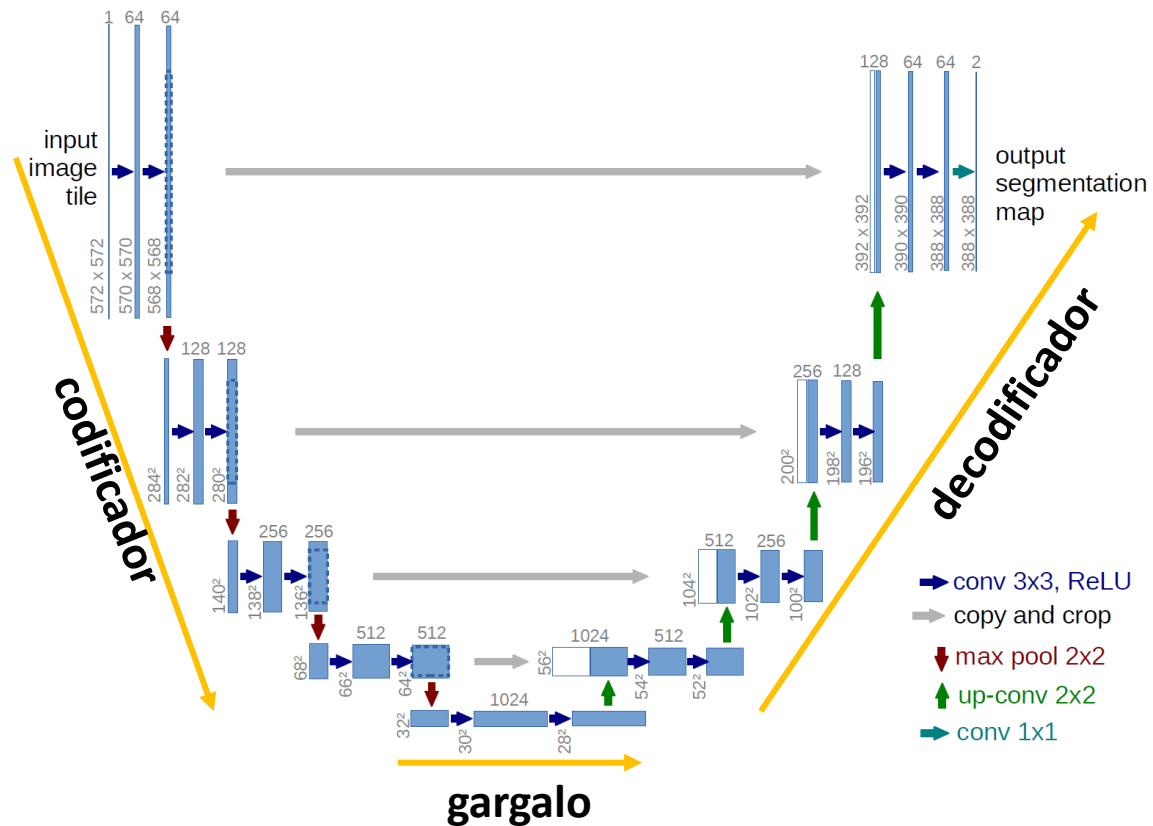
- Em seguida, o decodificador descompacta a representação latente de volta em uma imagem com a mesma resolução inicial.
- Ou seja, ele ***aumenta a resolução espacial***.

U-Net: Codificador



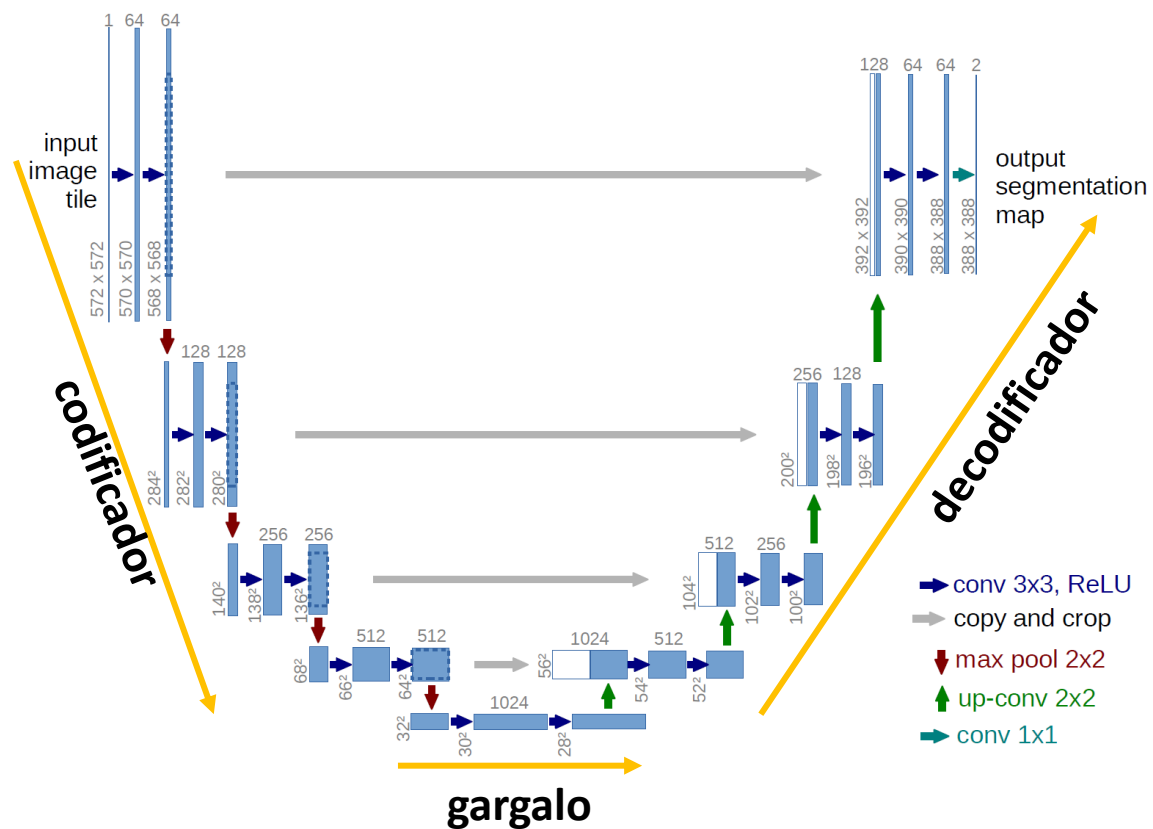
- O **codificador** é responsável por **identificar as características mais relevantes** na imagem de entrada.
- Suas camadas realizam operações **convoluções** (seguidas por ativação ReLU e *max pooling*).
- **As convoluções reduzem a resolução espacial** dos **mapas de características**, capturando assim representações cada vez mais abstratas da entrada.

U-Net: Codificador



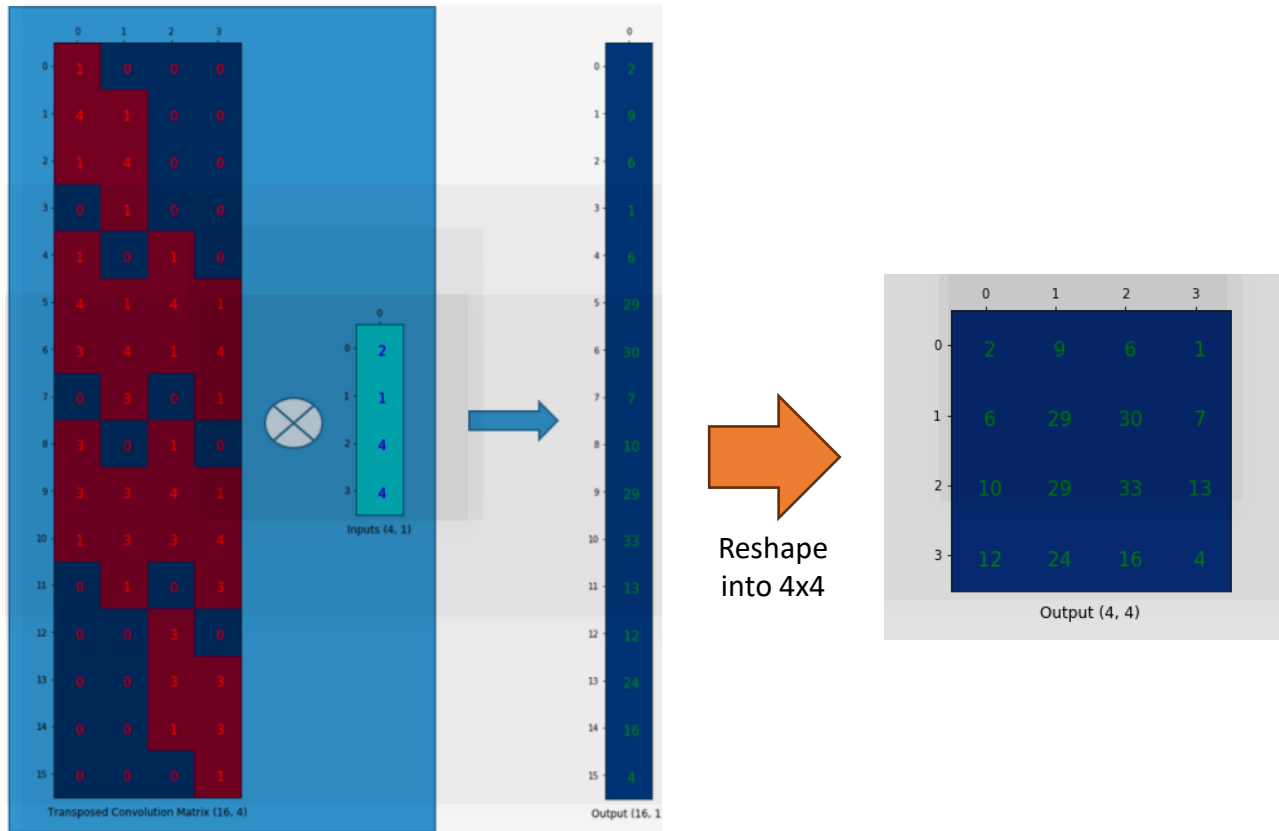
- Durante a **codificação**, a **informação espacial é reduzida** enquanto a **informação das características é aumentada** devido ao aumento do número de canais.
- A tarefa do codificador é semelhante à de outras CNNs.

U-Net: Decodificador



- O **decodificador** é responsável por decodificar (i.e., **descompactar**) as informações latentes capturadas pelo codificador, **mantendo a resolução espacial da entrada**.
- Suas camadas aumentam a resolução dos mapas de características através de **convoluções transpostas**.

U-Net: Decodificador

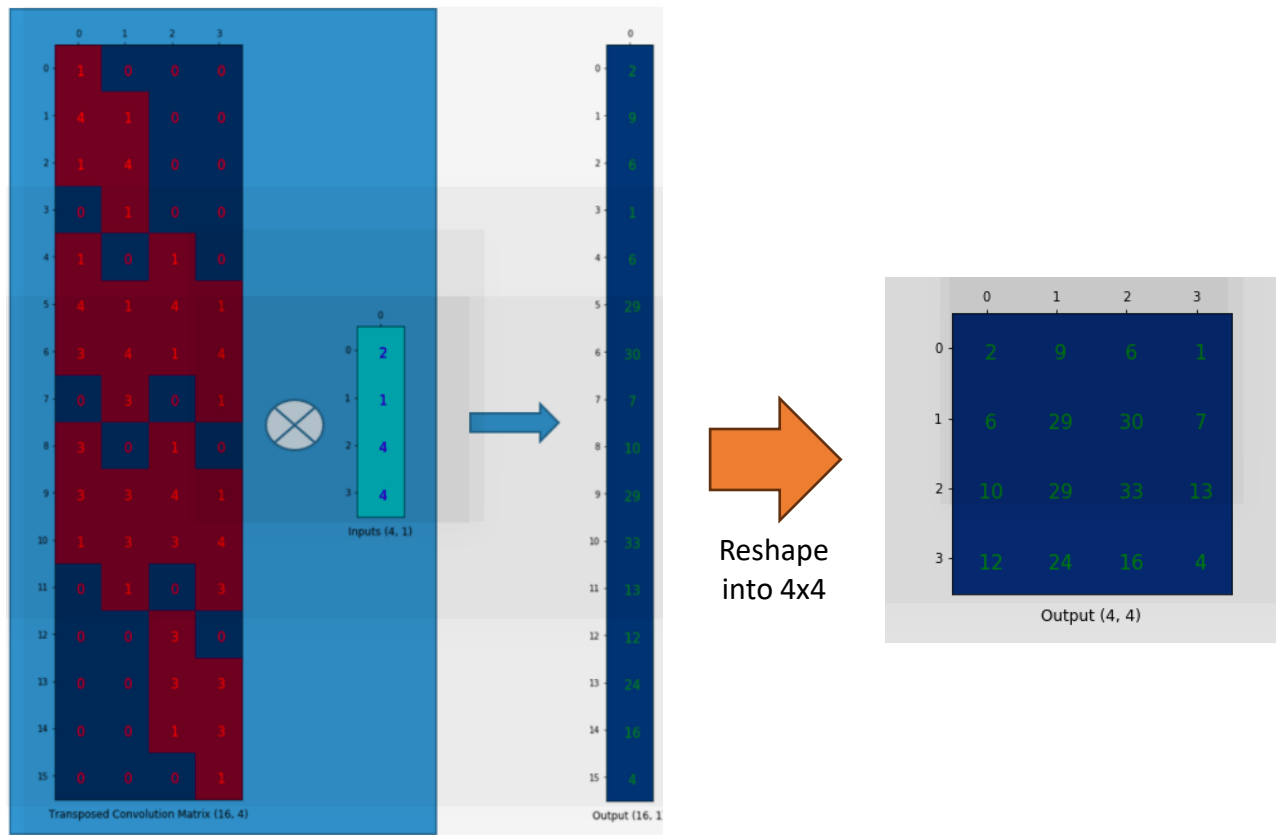


Matriz do
kernel
transposta

entrada

- Nas camadas de **convolução transposta**, as operações de *pooling* são substituídas por operações de **upsampling**.
- Ou seja, essas camadas aumentam a resolução da saída.

U-Net: Decodificador

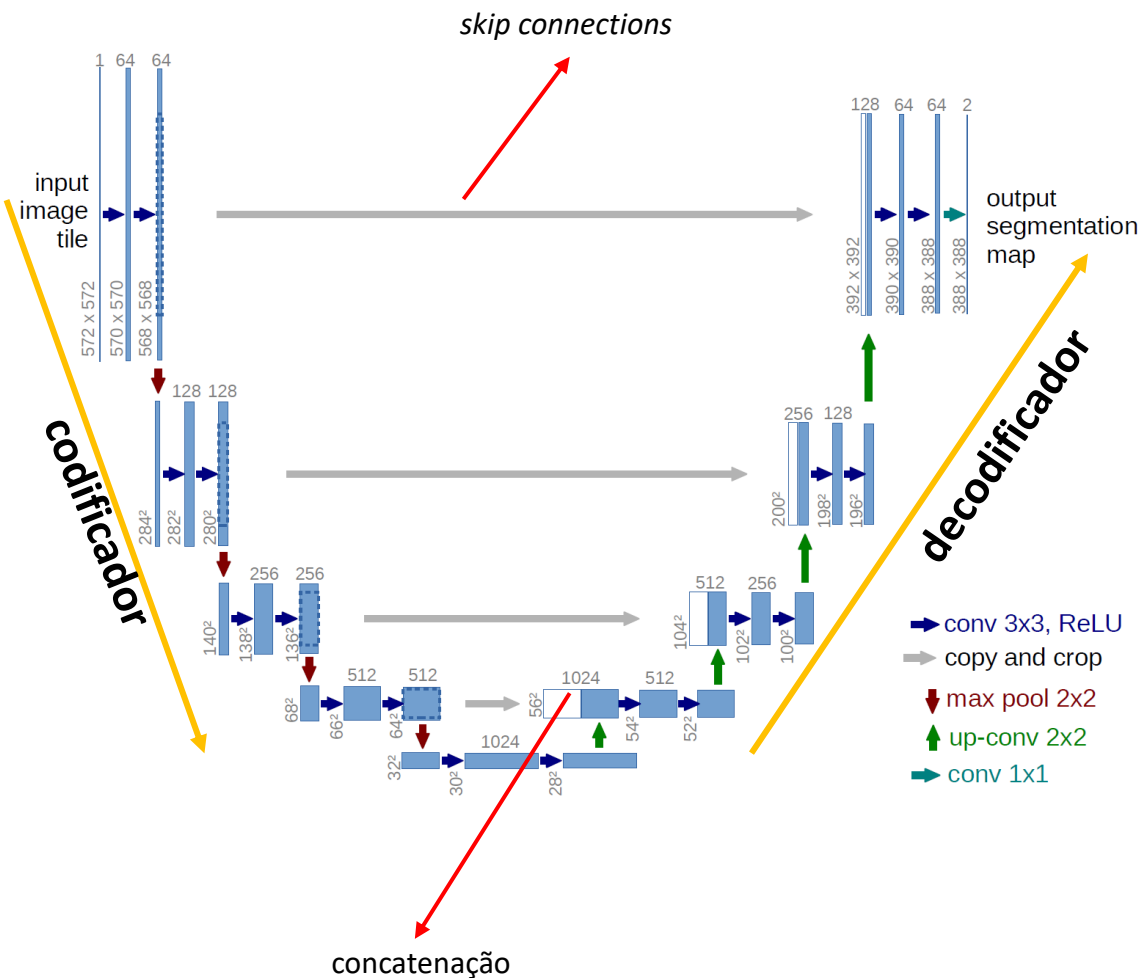


Matriz do
kernel
transposta

entrada

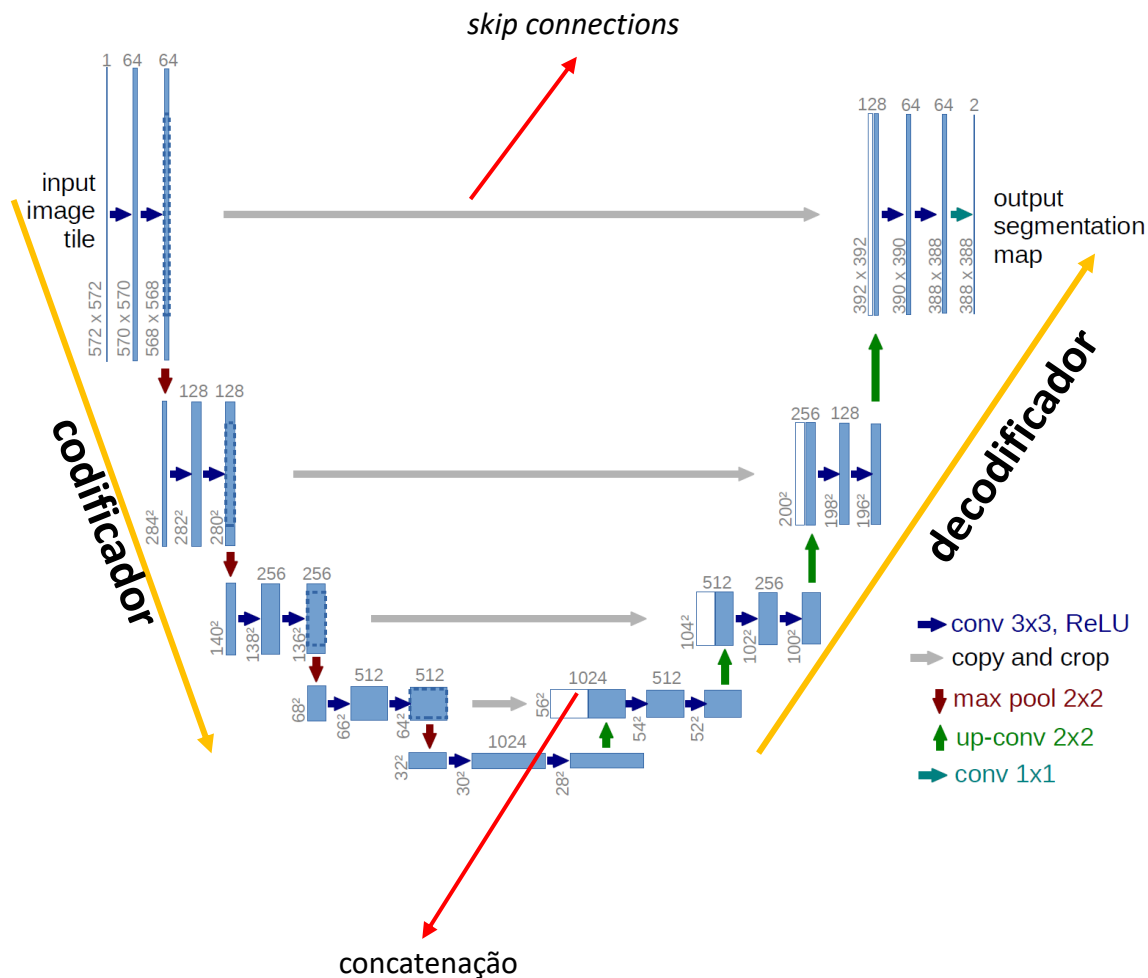
- A operação de convolução transposta é similar a de convolução, sendo a única diferença é que aqui *multiplicamos a matriz do kernel transposta pela entrada achatada e depois realizamos um reshape.*

U-Net: *Skip Connections*



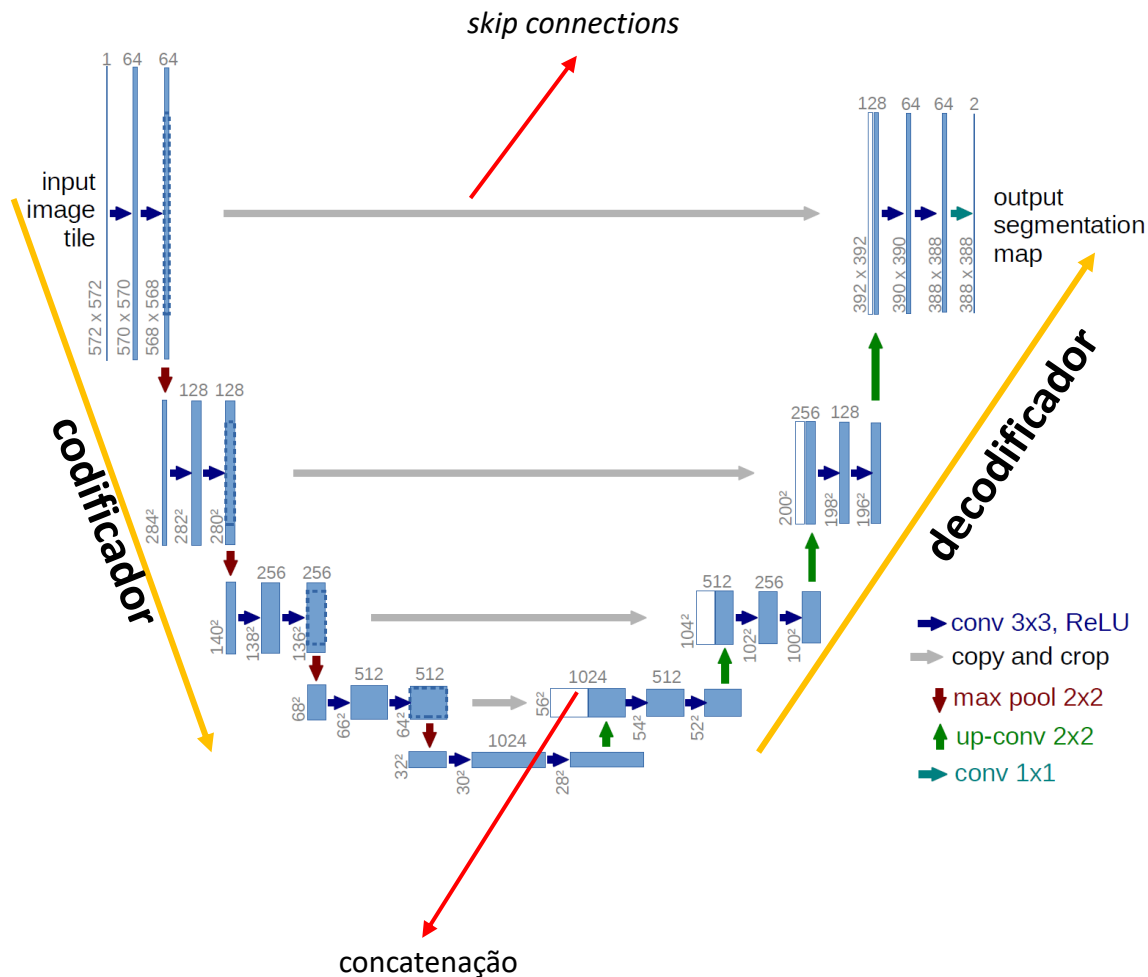
- CNNs, incluindo o codificador, tendem a *esquecer certas características ao longo das camadas convolucionais*.
- Para amenizar esse problema, usa-se *skip connections*.

U-Net: *Skip Connections*



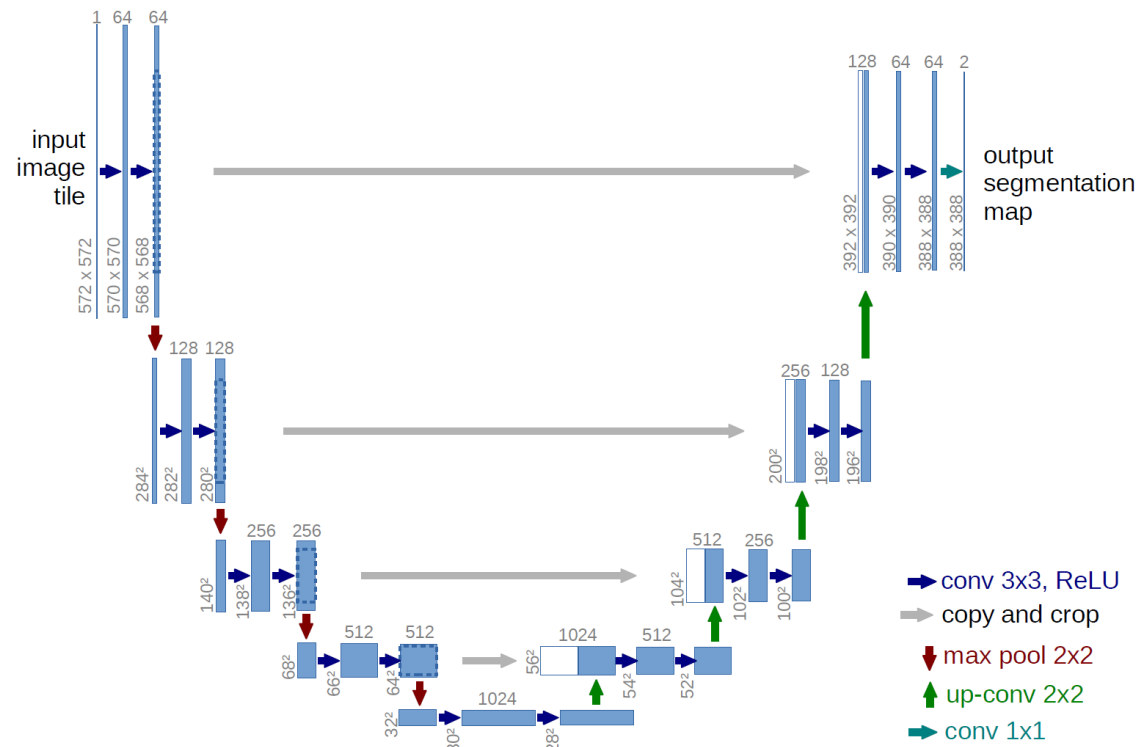
- As *skip connections* ajudam a *preservar* as *informações espaciais perdidas na codificação*.
- Isso ajuda as camadas decodificadoras a *localizar as características com mais precisão*.

U-Net: *Skip Connections*



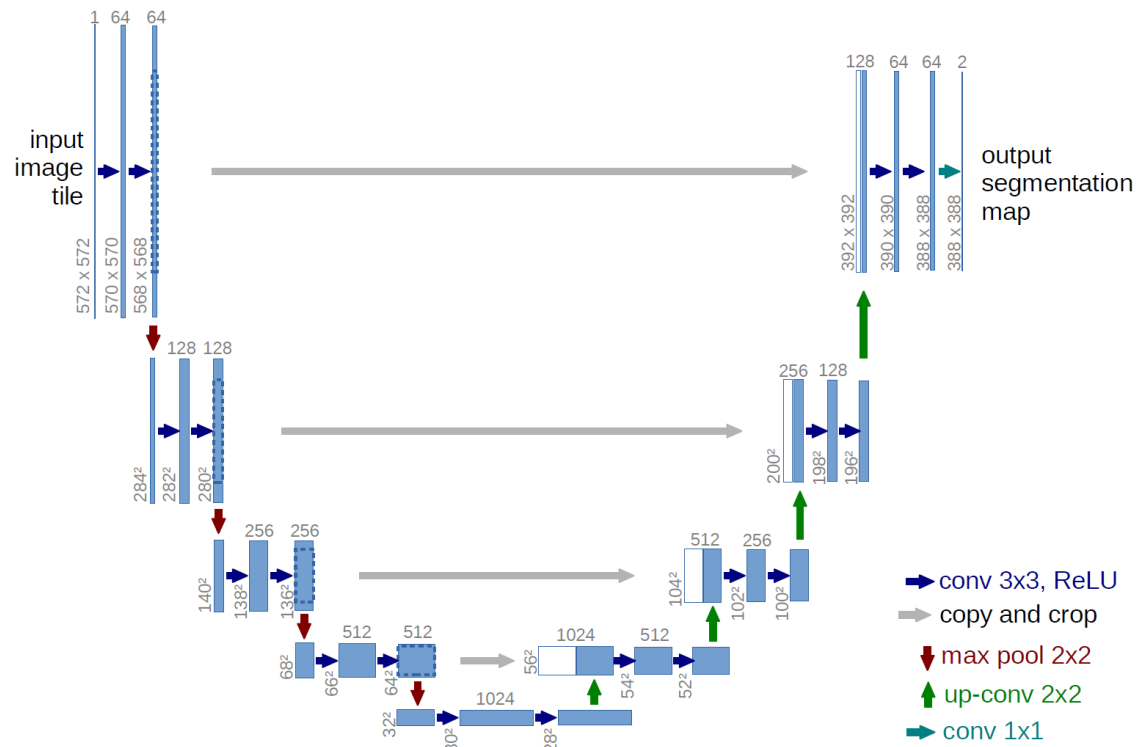
- Cada nível do decodificador **concatena** os **mapas de características locais** com **características de maior resolução** vindas do codificador através das **skip connections**.

U-Net: Informação temporal



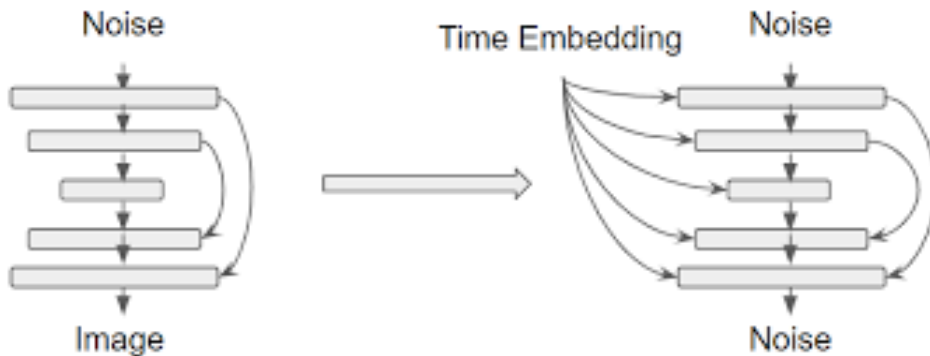
- Até o momento, vimos que a U-Net recebe um **tensor** com determinadas dimensões de entrada e gera um **tensor** com as mesmas dimensões em sua saída.

U-Net: Informação temporal



- Ou seja, ela recebe uma imagem e gera um tensor com a predição do ruído adicionado àquela imagem.
- Porém, lembrem-se que a rede deve receber também como entrada o passo, t (i.e., o nível de ruído) $\epsilon_{\theta}(x_t, t)$.

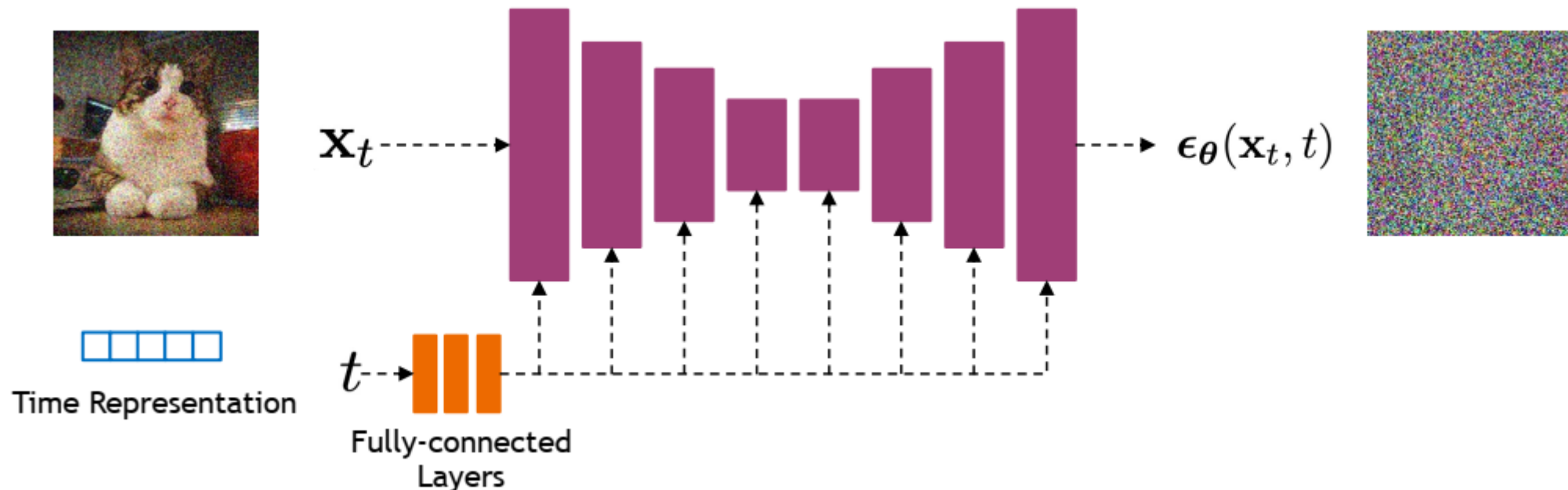
U-Net: Informação temporal



- A U-Net não sabe qual t estamos decodificando.
- Essa informação é crítica para o que modelo prediga com precisão o ruído em um determinado intervalo de tempo, t .
- Portanto, é necessário se modificar a arquitetura da *U-Net para que ela receba a informação temporal*.

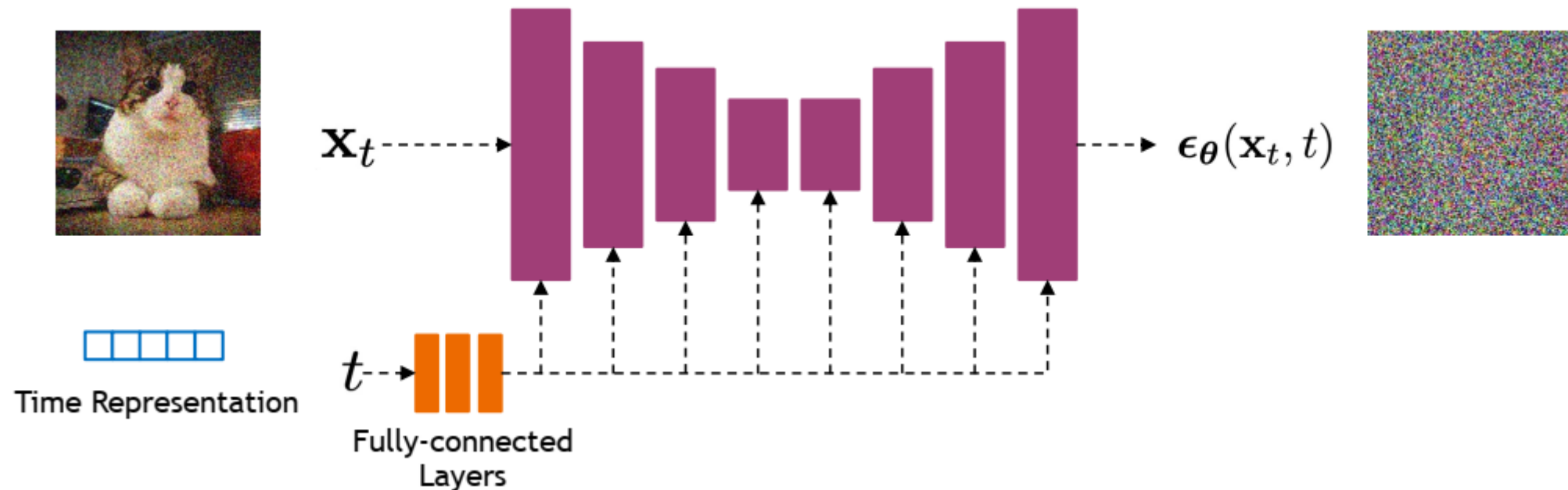
Positional embeddings

- Em geral, o passo de tempo t é **adicionado** como entrada através de **positional embeddings** a cada bloco de **down** e **upsampling**.
- Isso faz com que a rede neural “**saiba**” em qual **intervalo de tempo ela está operando e consiga prever o ruído**.



Positional embeddings

- Em geral, os **positional embeddings** são criados usando-se **camadas densas** (i.e., totalmente conectadas) e têm suas saídas **somadas** às entradas das camadas de **down** e **upsampling**.



Vantagens

- O **treinamento** de modelos de difusão é geralmente **mais estável** do que os de Redes Adversariais Generativas (GANs), que são notoriamente difíceis de treinar.
- Não sofrem com a **diversidade limitada de amostras** como as GANs.
 - GANs podem sofrer colapso de modo, onde o gerador produz amostras limitadas ou repetitivas.
- Mostraram-se **mais robustos ao overfitting** do que GANs.
- Na maioria dos casos, seu **desempenho é superior aos modelos generativos de última geração**, como GANs e Autoencoders Variacionais (VAEs).

Desvantagens

- Por ter que replicar a cadeia de Markov completa (de T a 0), o algoritmo de amostragem proposto em [1] é **lento na geração de novas amostras** em comparação com GANs.
 - Existem algumas propostas para superar essa desvantagem, como a do trabalho “Denoising Diffusion Implicit Models” [3], onde os autores substituíram a cadeia de Markov por um processo não Markoviano para amostrar mais rapidamente.
- São **computacionalmente intensivos** e requerem tempos de treinamento mais longos em comparação com GANs.
- Possuem **muitos hiperparâmetros** que devem ser ajustados para se obter bons resultados.

Aplicações



- Geração de vídeos a partir de *prompts* de texto.
- Para isso, o modelo precisa ser modificado para receber como entrada o texto, através de **word embeddings**.
- Exemplo: MagicVideo.

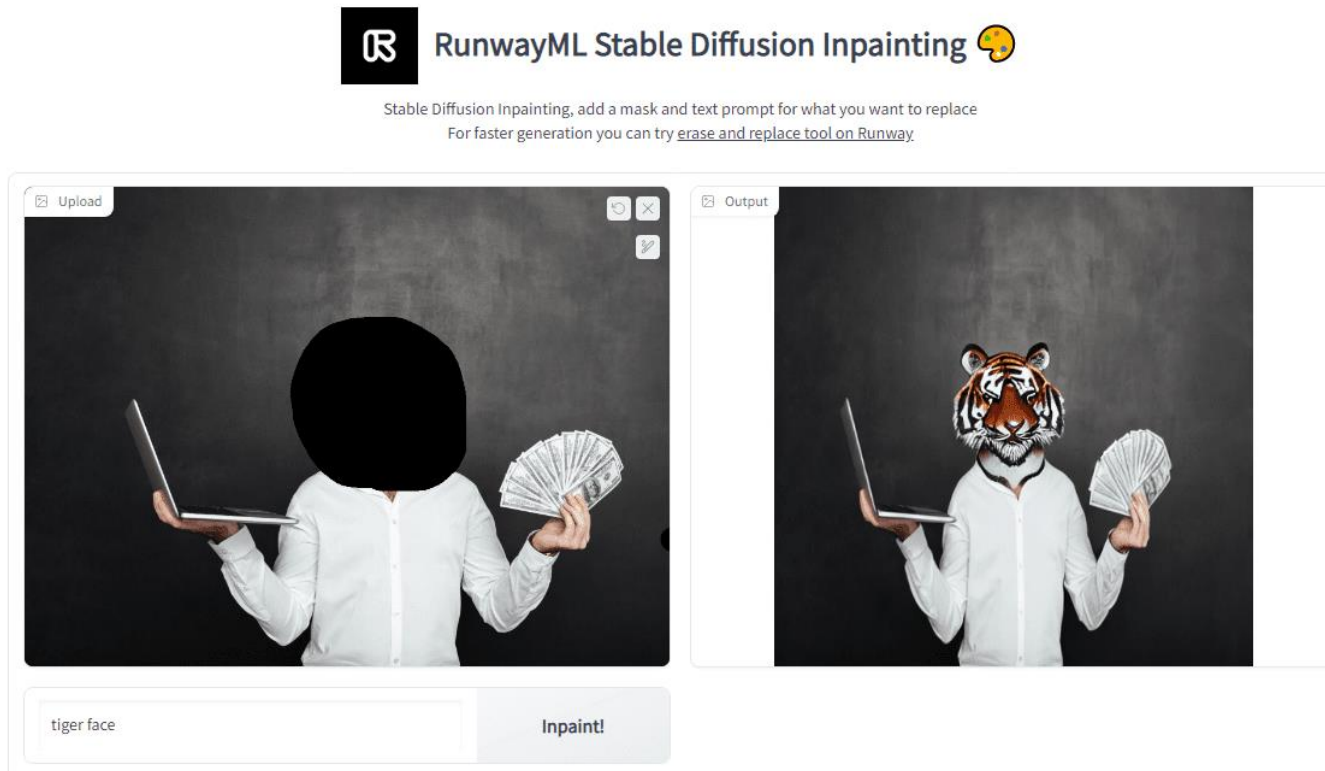
Aplicações



Picture of Chef Chopping Carrots on a cutting board

- Geração de imagens a partir de *prompts* de texto.
- Exemplos: Midjourney, Google Imagen, and DALL-E

Aplicações



- *Image inpainting* é uma técnica que **remove ou substitui elementos em imagens**.
- Exemplo: RunwayML.

Comparação com outros algoritmos

GAN



Diffusion Model



Dataset



- Modelos de difusão geram imagens mais diversificadas do que GANs [4].
- Por exemplo, na figura ao lado vemos
 - cabeças de avestruz ampliadas,
 - flamingos únicos,
 - diferentes orientações de cheeseburgers,
 - e um peixe sem nenhum humano segurando-o.

Comparação com outros algoritmos

Model	FID	sFID	Prec	Rec
LSUN Bedrooms 256×256				
DCTransformer [†] [42]	6.40	6.66	0.44	0.56
DDPM [25]	4.89	9.07	0.60	0.45
IDDPM [43]	4.24	8.21	0.62	0.46
StyleGAN [27]	2.35	6.62	0.59	0.48
ADM (dropout)	1.90	5.59	0.66	0.51

LSUN Horses 256×256				
StyleGAN2 [28]	3.84	6.46	0.63	0.48
ADM	2.95	5.94	0.69	0.55
ADM (dropout)	2.57	6.81	0.71	0.55

LSUN Cats 256×256				
DDPM [25]	17.1	12.4	0.53	0.48
StyleGAN2 [28]	7.25	6.33	0.58	0.43
ADM (dropout)	5.57	6.69	0.63	0.52

ImageNet 64×64				
BigGAN-deep* [5]	4.06	3.96	0.79	0.48
IDDPM [43]	2.92	3.79	0.74	0.62
ADM	2.61	3.77	0.73	0.63
ADM (dropout)	2.07	4.29	0.74	0.63

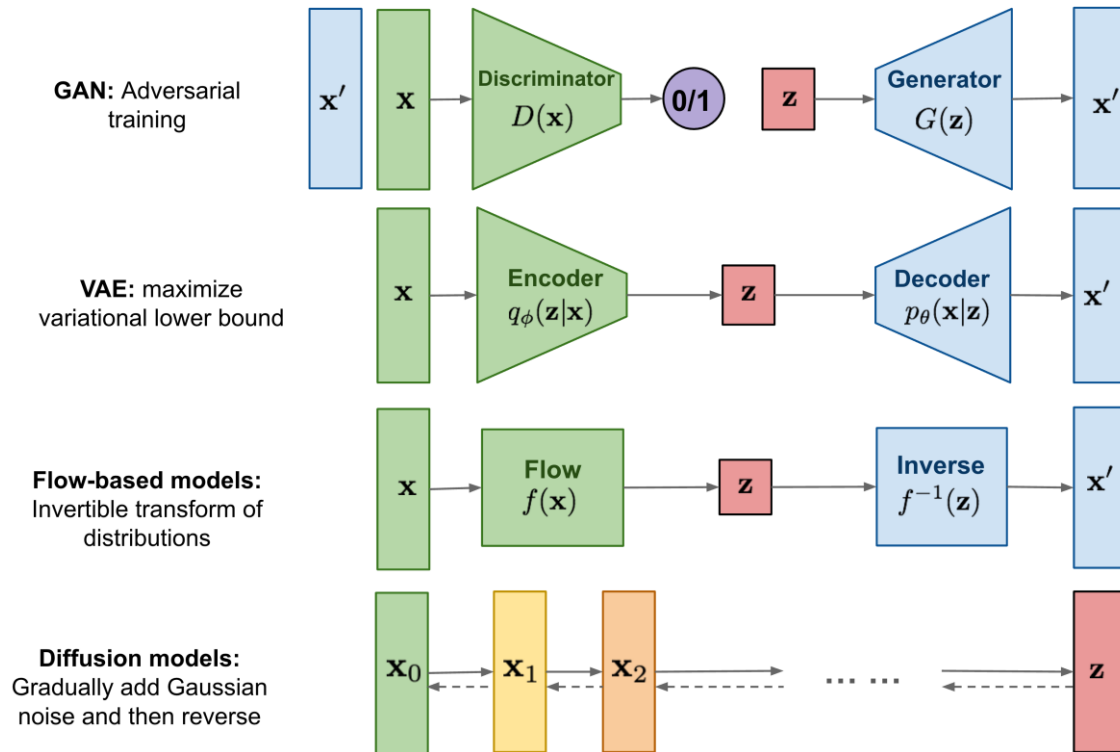
ImageNet 128×128				
BigGAN-deep [5]	6.02	7.18	0.86	0.35
LOGAN [†] [68]	3.36			
ADM	5.91	5.09	0.70	0.65
ADM-G (25 steps)	5.98	7.04	0.78	0.51
ADM-G	2.97	5.09	0.78	0.59

ImageNet 256×256				
DCTransformer [†] [42]	36.51	8.24	0.36	0.67
VQ-VAE-2 ^{††} [51]	31.11	17.38	0.36	0.57
IDDPM [†] [43]	12.26	5.42	0.70	0.62
SR3 ^{††} [53]	11.30			
BigGAN-deep [5]	6.95	7.36	0.87	0.28
ADM	10.94	6.02	0.69	0.63
ADM-G (25 steps)	5.44	5.32	0.81	0.49
ADM-G	4.59	5.25	0.82	0.52

ImageNet 512×512				
BigGAN-deep [5]	8.43	8.13	0.88	0.29
ADM	23.24	10.19	0.73	0.60
ADM-G (25 steps)	8.41	9.67	0.83	0.47
ADM-G	7.72	6.57	0.87	0.42

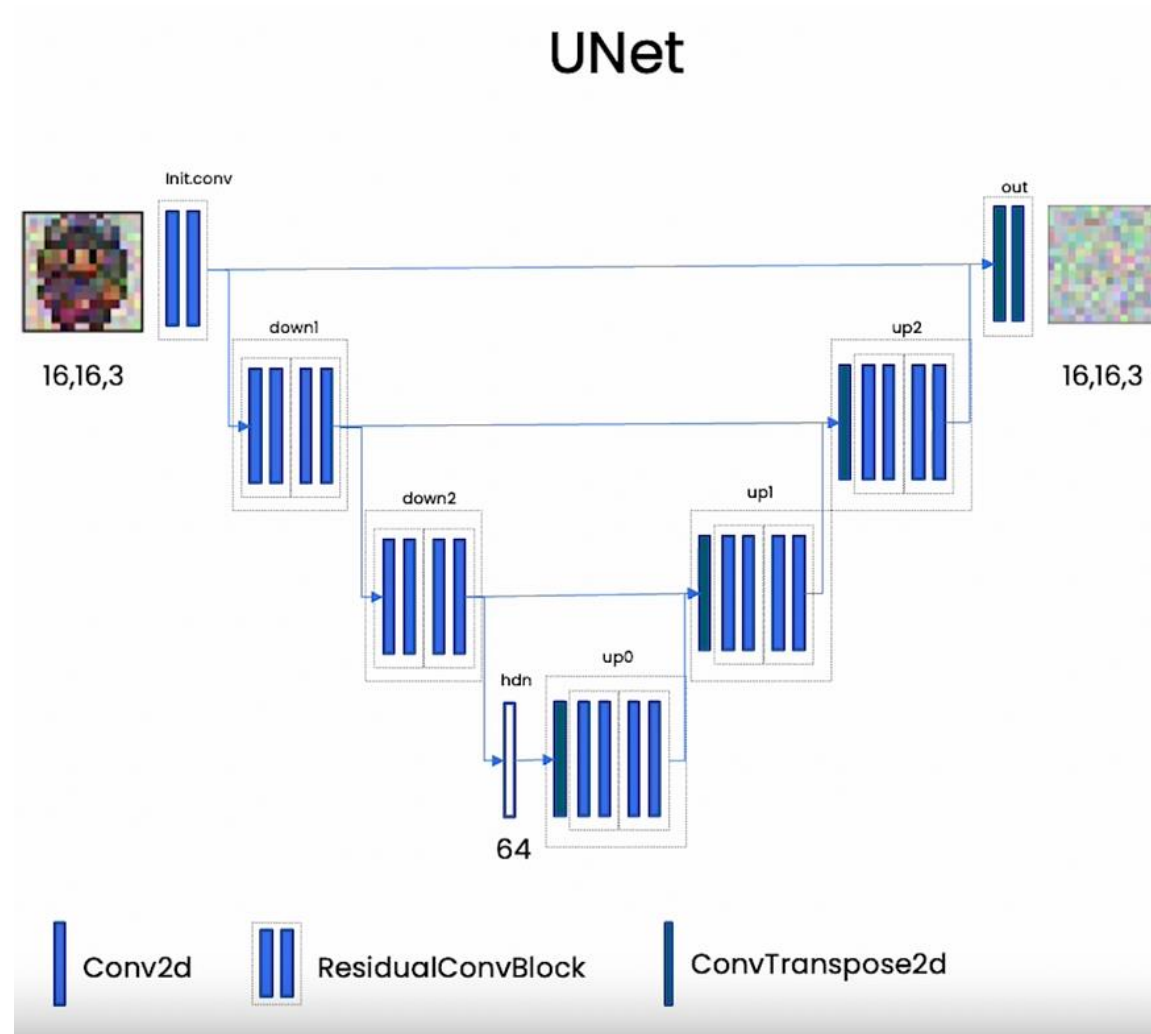
- O modelo de difusão obtém o melhor *Fréchet Inception Distance* (FID) em todas as tarefas executadas em [4].
 - FID é uma métrica usada para *capturar a diversidade das amostras*.
 - FID compara a distribuição das imagens geradas com a distribuição do conjunto de imagens reais ("ground Truth").

Outros modelos generativos



- Os modelos de difusão são apenas um entre os vários modelos generativos.
- Existem vários outros modelos com ideias diferentes por trás da geração das imagens sintéticas:
 - Generative Adversarial Networks (GANs).
 - Variational Autoencoders (VAEs).
 - Flow models.

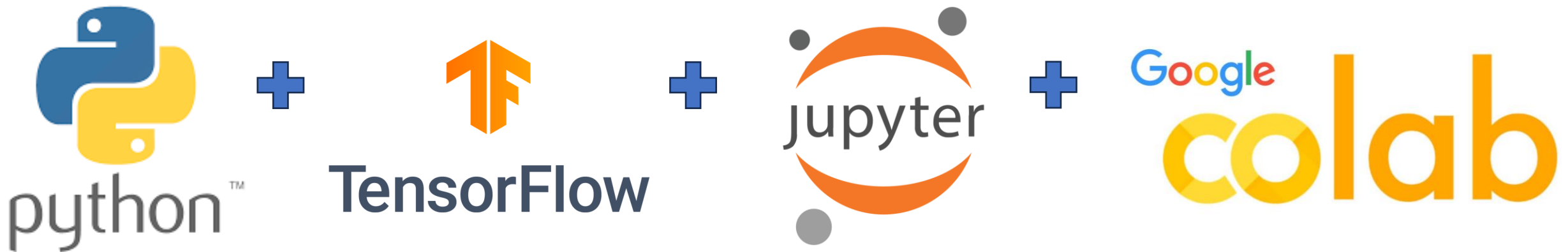
Exemplos



- Nos exemplos seguintes, usaremos uma U-Net com a arquitetura mostrada ao lado.

Exemplos

- [Sampling](#)
- [Training](#)



Quiz

- [Modelos de difusão](#)
- Anexem o *print screen* da pontuação obtida à tarefa específica do MS Teams.

Quiz - Diffusion Model

Total points 9/22

Referências

- [1] Jonathan Ho, et al., “Denoising Diffusion Probabilistic Models”, <https://arxiv.org/abs/2006.11239>
- [2] Olaf Ronneberger, et al., “U-Net: Convolutional Networks for Biomedical Image Segmentation”, <https://arxiv.org/abs/1505.04597>
- [3] Jiaming Song, Chenlin Meng, Stefano Ermon, “Denoising Diffusion Implicit Models”, <https://arxiv.org/abs/2010.02502>
- [4] Prafulla Dhariwal, Alex Nichol, “Diffusion Models Beat GANs on Image Synthesis”, <https://arxiv.org/abs/2105.05233>
- [5] Robin Rombach, et al., “High-Resolution Image Synthesis with Latent Diffusion Models”, <https://arxiv.org/abs/2112.10752>
- [6] Calvin Luo, “Understanding Diffusion Models: A Unified Perspective”, <https://arxiv.org/abs/2208.11970>

Perguntas?

Obrigado!

Anexo I

O truque de reparametrização:
Amostragem tratável em forma
fechada em qualquer passo de
tempo

O truque de reparametrização

- Usando o **truque da reparametrização**, podemos amostrar em qualquer instante de tempo, t , sempre começando de \mathbf{x}_0 (i.e., a imagem original).
- Vamos começar pelo instante, t .

$$\mathbf{x}_t = \sqrt{1 - \beta_t} \mathbf{x}_{t-1} + \sqrt{\beta_t} \boldsymbol{\epsilon}_{t-1}.$$

- Definindo $\alpha_t = 1 - \beta_t$, podemos reescrever \mathbf{x}_t como

$$\mathbf{x}_t = \sqrt{\alpha_t} \mathbf{x}_{t-1} + \sqrt{1 - \alpha_t} \boldsymbol{\epsilon}_{t-1}.$$

- Na sequência, definimos \mathbf{x}_{t-1} da mesma forma

$$\mathbf{x}_{t-1} = \sqrt{\alpha_{t-1}} \mathbf{x}_{t-2} + \sqrt{1 - \alpha_{t-1}} \boldsymbol{\epsilon}_{t-2}.$$

- Substituindo \mathbf{x}_{t-1} em \mathbf{x}_t temos

$$\mathbf{x}_t = \sqrt{\alpha_t} \{ \sqrt{\alpha_{t-1}} \mathbf{x}_{t-2} + \sqrt{1 - \alpha_{t-1}} \boldsymbol{\epsilon}_{t-2} \} + \sqrt{1 - \alpha_t} \boldsymbol{\epsilon}_{t-1}.$$

O truque de reparametrização

- Organizando os termos temos

$$\mathbf{x}_t = \sqrt{\alpha_t \alpha_{t-1}} \mathbf{x}_{t-2} + \sqrt{\alpha_t (1 - \alpha_{t-1})} \boldsymbol{\epsilon}_{t-2} + \sqrt{1 - \alpha_t} \boldsymbol{\epsilon}_{t-1}.$$

- A soma de duas variáveis aleatórias normalmente distribuídas é outra variável aleatória normalmente distribuída com média e variância iguais à soma das médias e variâncias das variáveis aleatórias individuais.

- Portanto,

$$\mathbf{x}_t = \sqrt{\alpha_t \alpha_{t-1}} \mathbf{x}_{t-2} + \sqrt{\alpha_t (1 - \alpha_{t-1}) + 1 - \alpha_t} \boldsymbol{\epsilon}_{t-2}.$$

- Simplificando o segundo termo, temos

$$\mathbf{x}_t = \sqrt{\alpha_t \alpha_{t-1}} \mathbf{x}_{t-2} + \sqrt{1 - \alpha_t \alpha_{t-1}} \boldsymbol{\epsilon}_{t-2}.$$

O truque de reparametrização

- Substituindo \mathbf{x}_{t-2} em \mathbf{x}_t , temos

$$\mathbf{x}_t = \sqrt{\alpha_t \alpha_{t-1} \alpha_{t-2}} \mathbf{x}_{t-3} + \sqrt{1 - \alpha_t \alpha_{t-1} \alpha_{t-2}} \boldsymbol{\epsilon}_{t-3}.$$

- Se prosseguirmos até \mathbf{x}_0 , temos

$$\mathbf{x}_t = \sqrt{\alpha_t \alpha_{t-1} \alpha_{t-2} \dots \alpha_1} \mathbf{x}_0 + \sqrt{1 - \alpha_t \alpha_{t-1} \alpha_{t-2} \dots \alpha_1} \boldsymbol{\epsilon}_0.$$

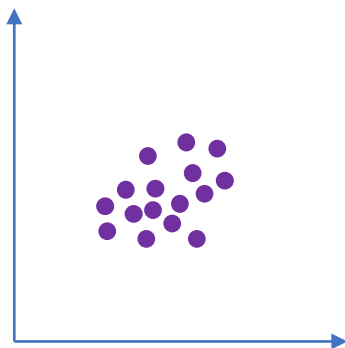
- Portanto, definindo $\bar{\alpha}_t = \alpha_t \alpha_{t-1} \alpha_{t-2} \dots \alpha_1 = \prod_{i=1}^t \alpha_i$, podemos reescrever \mathbf{x}_t como

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon},$$

onde $\boldsymbol{\epsilon}_t, \boldsymbol{\epsilon}_{t-1}, \dots, \boldsymbol{\epsilon}_0, \boldsymbol{\epsilon} \sim N(\mathbf{0}, \mathbf{I})$.

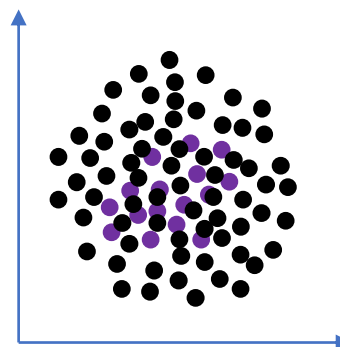
Figuras

Distribuição das imagens que
ocorrem naturalmente



● Imagem no espaço de imagens

Distribuição das imagens que
ocorrem naturalmente + ruído



● Imagem no espaço de imagens

● Imagem no espaço de imagens + ruído