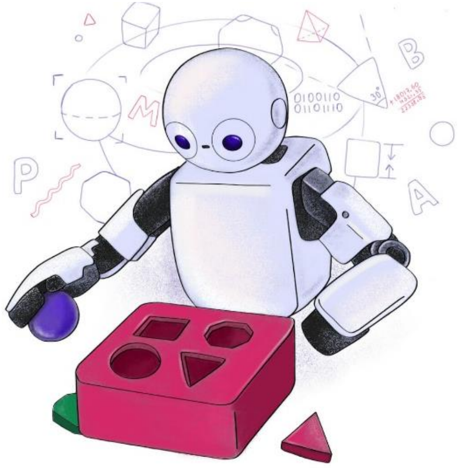
TP558 - Tópicos avançados em Machine Learning:

Diffusion Models



Inatel

Felipe Augusto Pereira de Figueiredo
felipe.figueiredo@inatel.br

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.
  - Alguns geram imagens com base em instruções (chamados de *prompts* em inglês).
- Eles também são usados para gerar vídeos, música, novas drogas, etc.

Cherry Blossom near a lake, snowing

Alone astronaut on Mars, mysterious, colorful, hyper realistic

Introdução

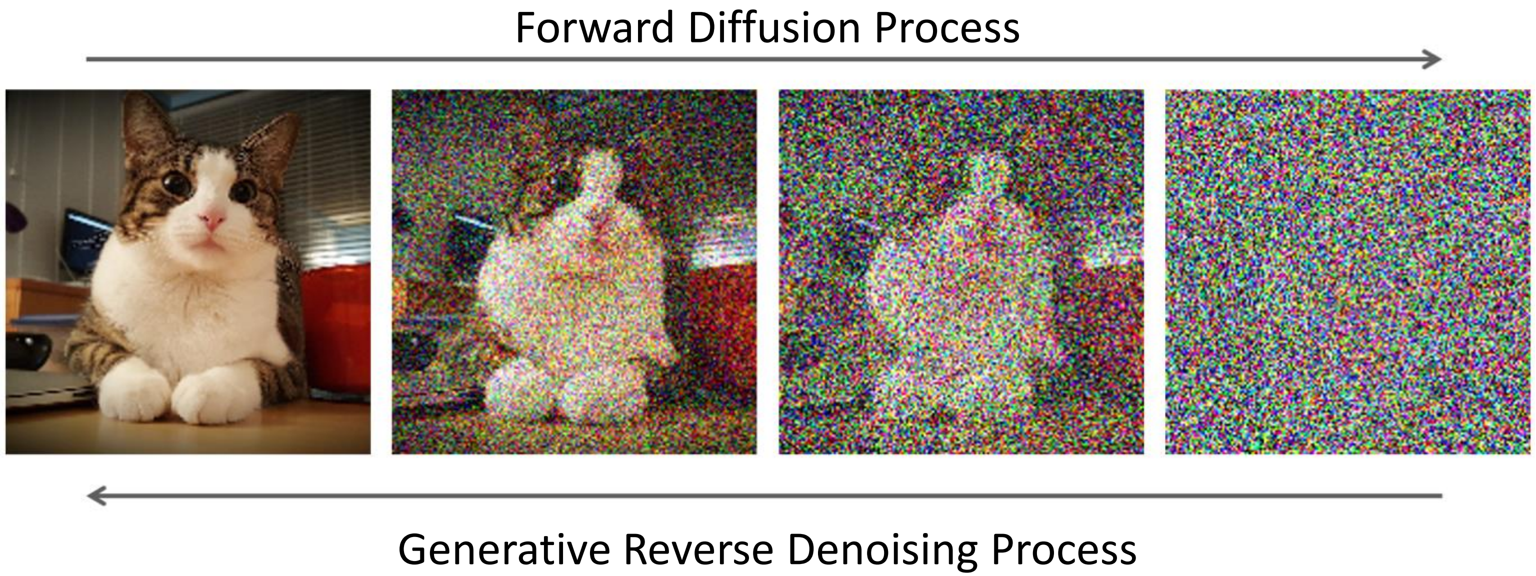
- Eles recebem esse nome devido à sua ***semelhança*** com o ***processo de difusão***, que descreve como partículas ou moléculas se movem de uma área de alta concentração para uma de baixa concentração impulsionadas pelo movimento térmico aleatório das partículas.

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).

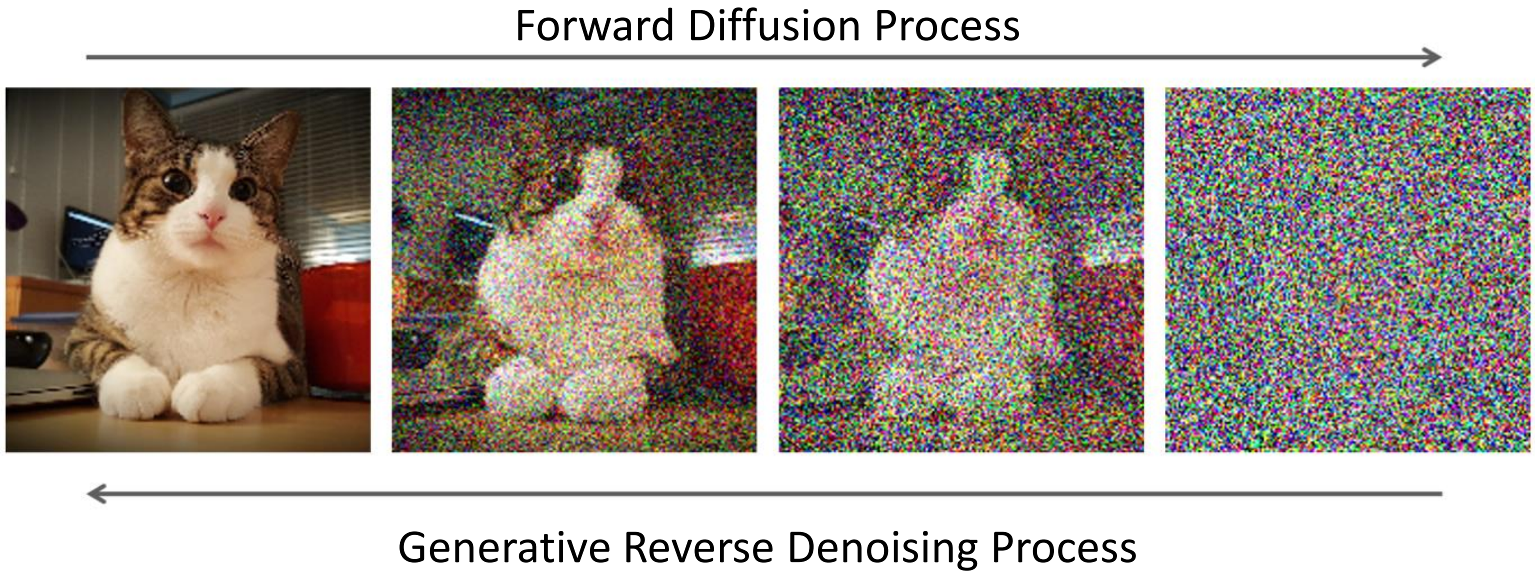
Introdução

- 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 e podem gerar imagens eliminando iterativamente o ruído de sinais puramente ruidosos.
- Portanto, neste seminário, nós veremos como esses modelos funcionam.

Termodinâmica de não equilíbrio

Distribuição das imagens que ocorrem naturalmente

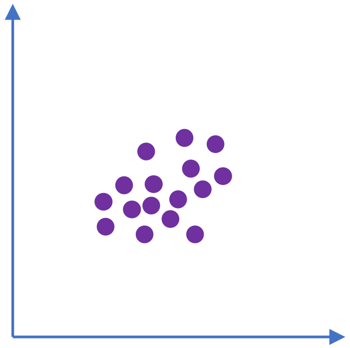
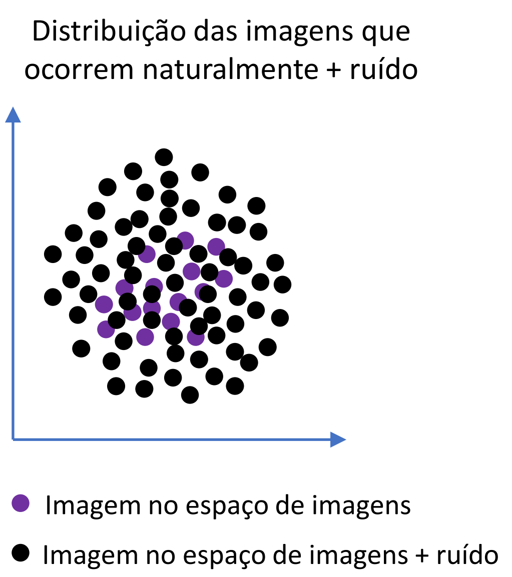


Imagem no espaço de imagens

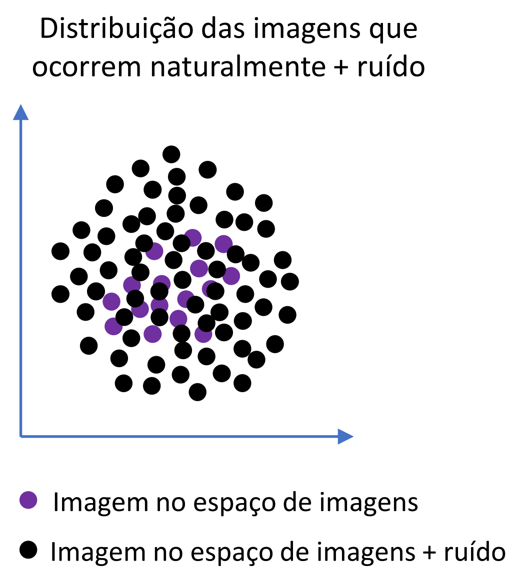
- Vamos considerar, por exemplo, a distribuição, q, de todas as imagens que ocorrem naturalmente.
  - A distribuição q é chamada de ***distribuição original***, alvo ou objetivo.
- Cada imagem é um ponto no espaço criado por todas as imagens e a distribuição das imagens que ocorrem naturalmente é uma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

Termodinâmica de não equilíbrio



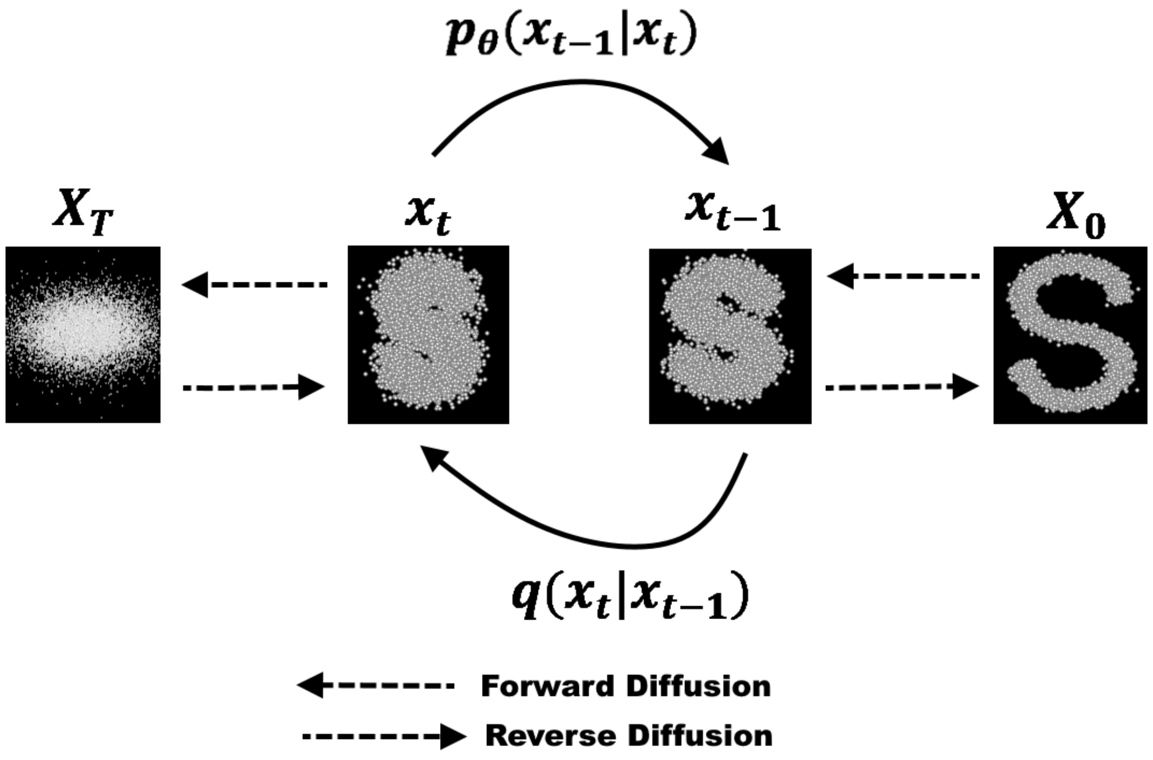
- Ao adicionar repetidamente ruído às imagens, a distribuição se ***difunde*** para o resto do espaço de imagens, até que a nuvem torna-se praticamente ***indistinguível de uma distribuição gaussiana***, $N(\mathbf{0}, \boldsymbol{I})$.
  - N em geral é uma distribuição multidimensional, onde $\mathbf{0}$ é o vetor de médias e $\boldsymbol{I}$ é a matriz identidade.
- Um modelo que pode desfazer aproximadamente a ***difusão*** pode então ser usado para extrair amostras da distribuição original, q.
- Portanto, o modelo remove o ruído de uma amostra até que reste apenas valores retirados de distribuição original, q.

Termodinâmica de não equilíbrio



- 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.
- A distribuição inicial estando muito 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 gera a distribuição de probabilidade de um determinado conjunto de dados a partir apenas de ruído.

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 um amostra da distribuição alvo.
- 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***.

Processo de difusão direta

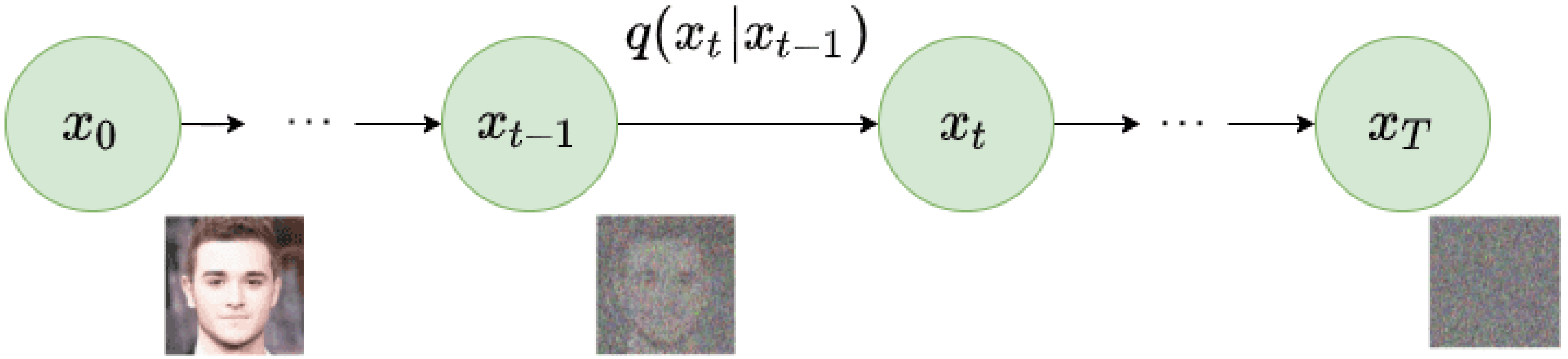
- Dado uma amostra de dados retirada de uma distribuição de dados real, $\boldsymbol{x}_0 \sim q(\boldsymbol{x})$, vamos definir um processo de ***difusão direta*** no qual adicionamos uma pequena quantidade de ruído gaussiano à amostra em T ***passos***, produzindo uma sequência de amostras ruidosas $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$.
  - $q(\boldsymbol{x})$ é a distribuição de probabilidade a ser aprendida.
- 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 varia ao longo do tempo produz resultados melhores: variação linear, quadrática, cossenoidal, etc.
- A variação do valor de β_t garante que $\boldsymbol{x}_T$ tenha praticamente uma distribuição Gaussiana isotrópica para T suficientemente grande.

Processo de difusão direta

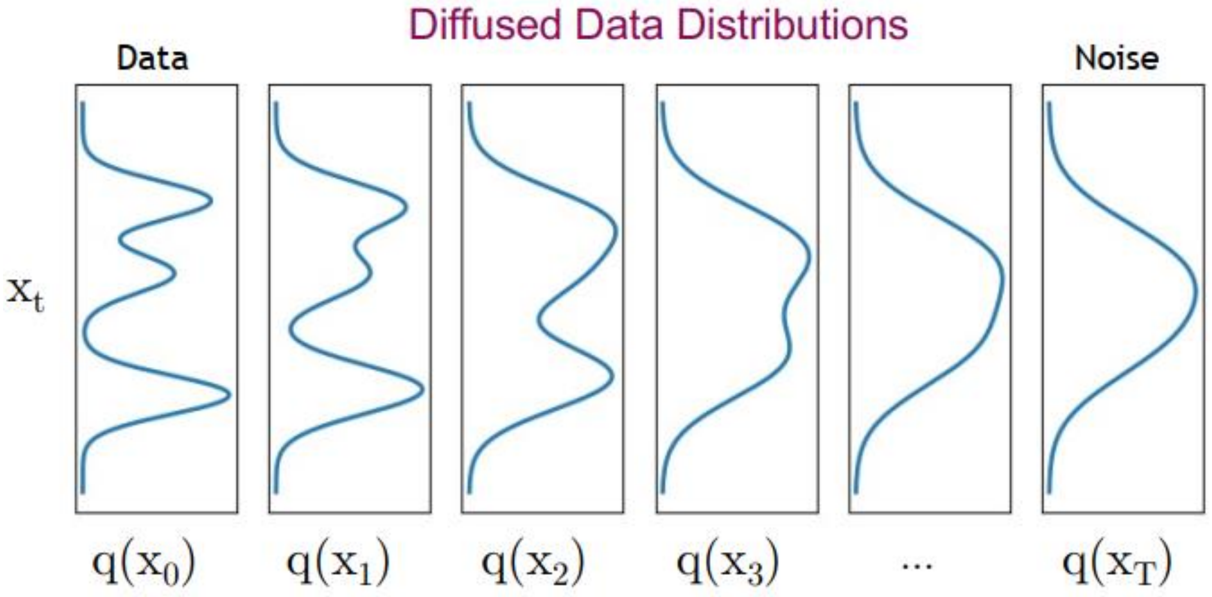
- Em cada passo, t, da cadeia de Markov adicionamos ruído gaussiano com variância β_t à $\boldsymbol{x}_{t-1}$, gerando uma nova variável $\boldsymbol{x}_t$ com ***distribuição condicional***

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \sqrt{1-\beta_t}\boldsymbol{x}_{t-1} + \sqrt{\beta_t}\boldsymbol{\epsilon} = N\left(\boldsymbol{x}_t; \boldsymbol{\mu}_t = \sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \boldsymbol{\Sigma}_t = \beta_t \boldsymbol{I}\right),$$

onde $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \boldsymbol{I})$.



Processo de difusão direta



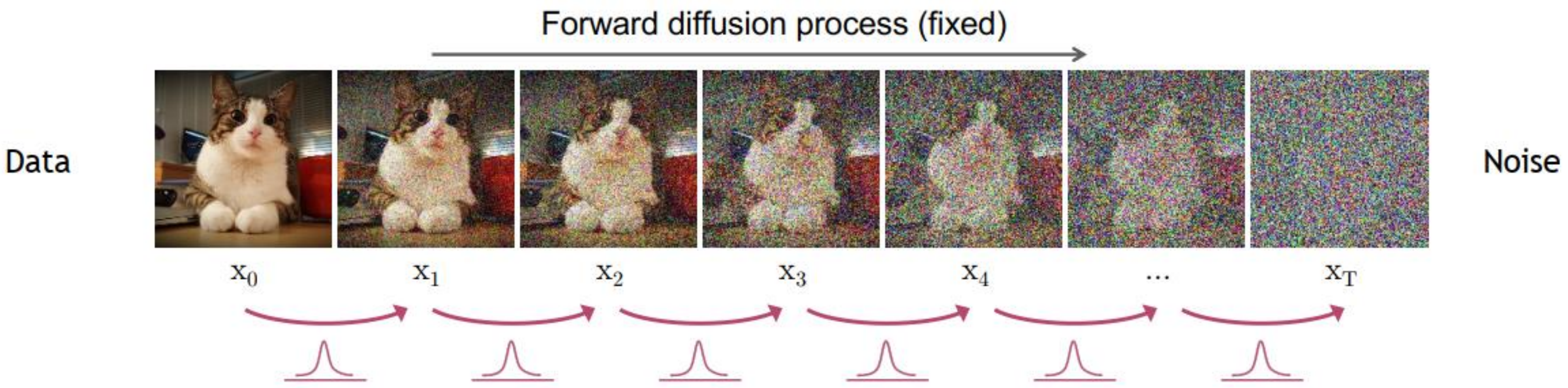
- A amostra de dados, $\boldsymbol{x}_0$, perde gradualmente suas características distinguíveis à medida que o passo se torna maior.
- Eventualmente, quando $T \rightarrow \infty$, $\boldsymbol{x}_T$ é equivalente a uma distribuição Gaussiana isotrópica.

Processo de difusão direta

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

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

- Esse processo que vai de $\boldsymbol{x}_0$ até $\boldsymbol{x}_T$ é chamado de ***trajetória***.



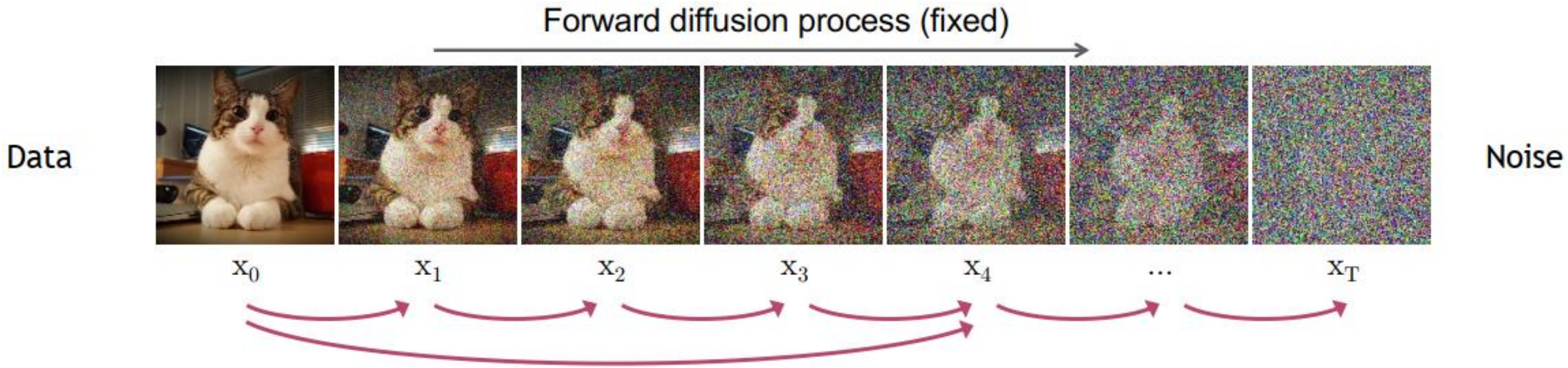
Processo de difusão direta

- Usando um truque de reparametrização, podemos amostrar $\boldsymbol{x}_t$ em ***qualquer instante de tempo***, t, como

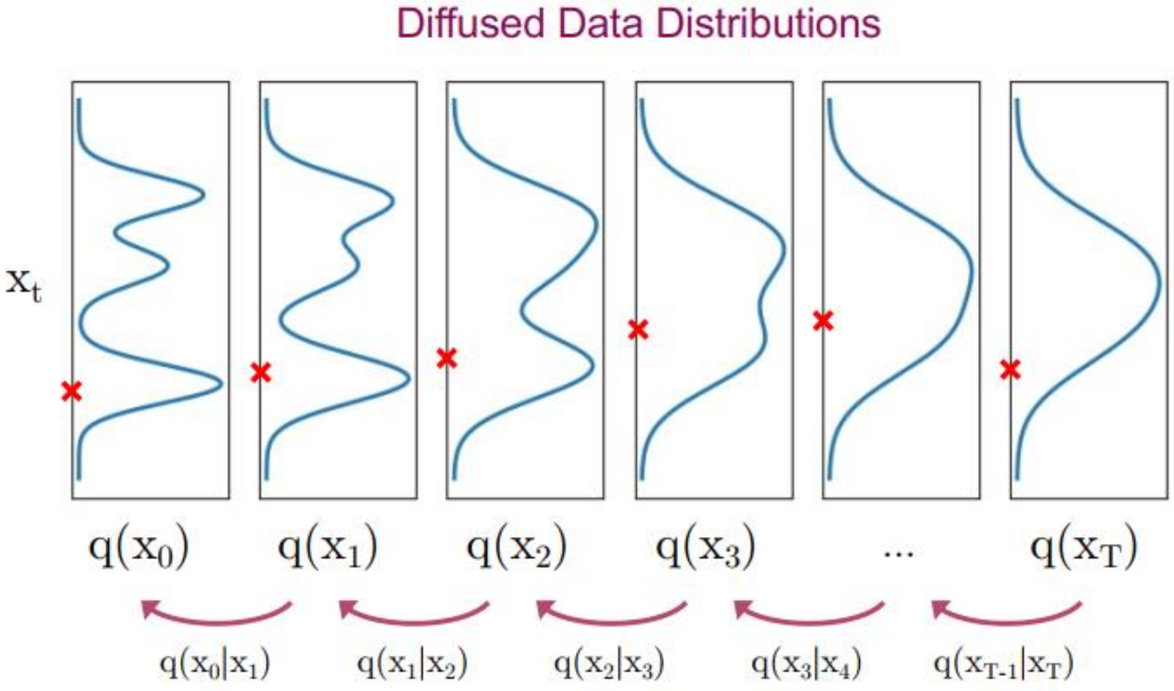
$$q(\boldsymbol{x}_t|\boldsymbol{x}_0) = N\left(\boldsymbol{x}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1-\bar{\alpha}_t)\boldsymbol{I}\right),$$

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

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



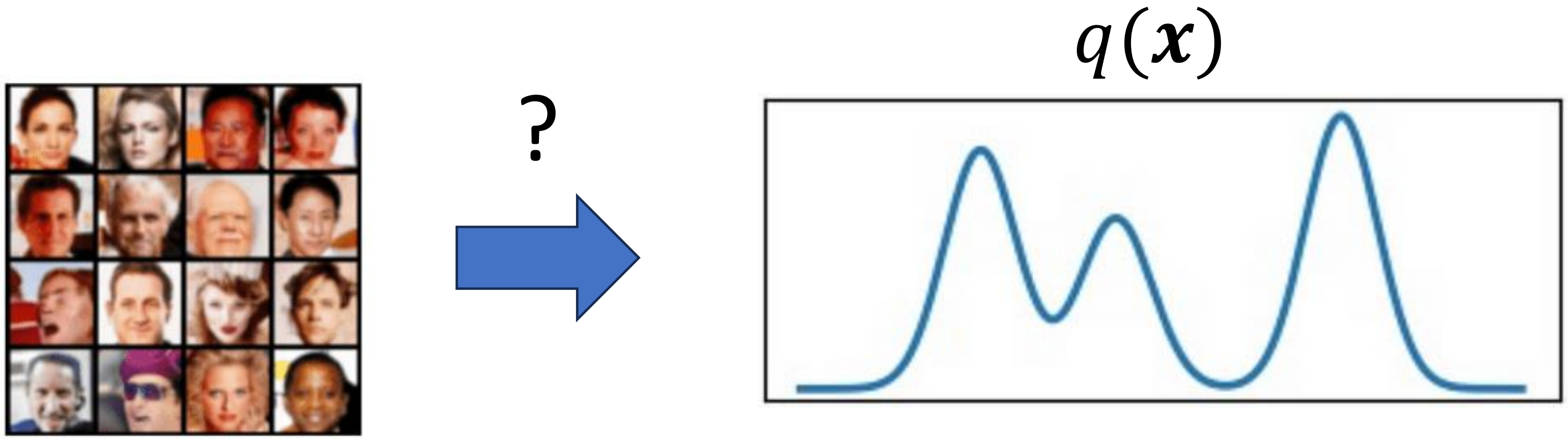
Processo de difusão reversa



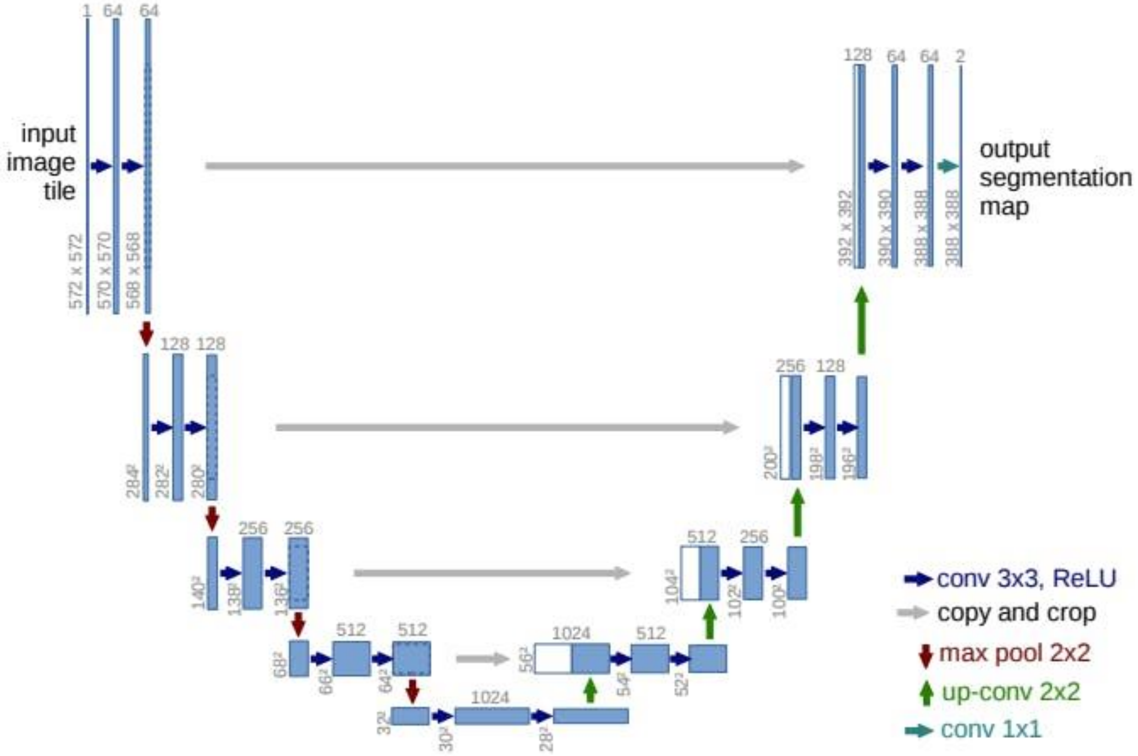
- A “mágica” dos modelos de difusão ocorre no processo inverso.
- Se pudermos reverter o processo direto e amostrar da distribuição $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$, poderemos recriar a amostra verdadeira, $\boldsymbol{x}_0$, a partir de uma amostra de ruído Gaussiano, $\boldsymbol{x}_T \sim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

Processo de difusão reversa

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



Processo de difusão reversa



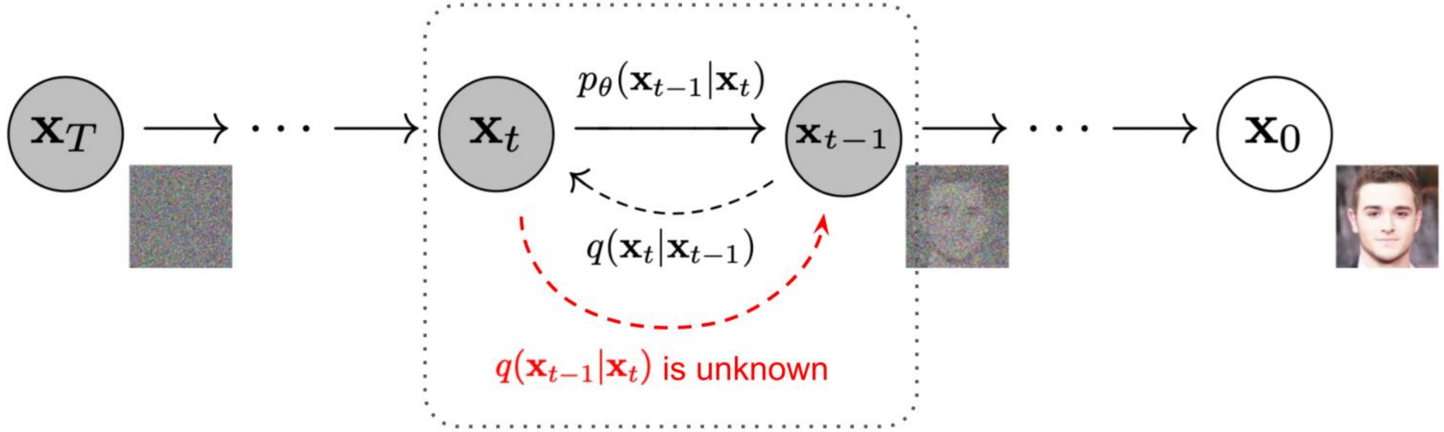
- Então como podemos modelar o processo de difusão reversa?
- Nós aproximamos o processo reverso, $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$, com um modelo parametrizável, p_θ, (e.g., uma rede neural), onde θ são os parâmetros da rede neural, atualizados pelo gradiente descendente.

Processo de difusão reversa

- Como $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ pode ser assumido Gaussiano se β_t for pequeno o suficiente em cada passo da difusão direta, podemos escolher p_θ como sendo uma distribuição Gaussiana e apenas parametrizar sua média e variância

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

- **OBS.**: A média e a variância também estão condicionadas ao passo, t, o qual dita o nível de ruído.



Processo de difusão reversa

- A rede recebe dois argumentos, $\boldsymbol{x}_t$ e t, e gera em sua saída um vetor $\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t)$ e uma matriz $\boldsymbol{\Sigma}_\theta(\boldsymbol{x}_t, t)$, de modo que cada passo do processo de difusão direta possa ser aproximadamente desfeita por
$$\boldsymbol{x}_{t-1} \sim N(\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t), \boldsymbol{\Sigma}_\theta(\boldsymbol{x}_t, t)).$$
- Começando com ruído puro, $p(\boldsymbol{x}_T) = N(\boldsymbol{x}_T; \boldsymbol{0}, \boldsymbol{I})$, o modelo aprende a distribuição conjunta como
$$p_\theta(\boldsymbol{x}_{0:T}) = p(\boldsymbol{x}_T) \prod_{t=1}^{T} p_\theta(\boldsymbol{x}_{t-1} | \boldsymbol{x}_t) = p(\boldsymbol{x}_T) \prod_{t=1}^{T} N(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t), \boldsymbol{\Sigma}_\theta(\boldsymbol{x}_t, t)).$$
- A distribuição $p_\theta(\boldsymbol{x}_{0:T})$ também é chamada de ***trajetória***.

Processo de difusão reversa

- Os autores do artigo DDPM [1], fazem uma simplificação onde $\boldsymbol{\Sigma}_\theta(\boldsymbol{x}_t, t) = \sigma_t^2 \boldsymbol{I}$, onde $\sigma_t^2 = \beta_t$ ou $\sigma_t^2 = \tilde{\beta}_t$, onde $\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$.
- Desta forma, a rede neural precisa aprender apenas a média $\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t)$ da distribuição condicional de probabilidade, $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$.

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \underbrace{\mu_\theta(\mathbf{x}_t, t)}_{\text{Rede neural (e.g., U-net)}}, \sigma_t^2\mathbf{I})$$

- Agora veremos como o modelo é treinado.

Treinando um modelo de difusão

- O treinamento é realizado minimizando-se o ***limite superior variacional*** da probabilidade logarítmica negativa

Função de perda

$$\mathrm{E}[-\log p_\theta(\boldsymbol{x}_0)] \leq \mathrm{E}_q\left[-\log \frac{p_\theta(\boldsymbol{x}_{0:T})}{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\right] = L.$$

- Os autores de [1] mostram que L pode ser reescrito como uma ***soma de perdas***

$$\begin{aligned} L &= \mathrm{E}_q\Bigg[\underbrace{D_{KL}(q(\boldsymbol{x}_T|\boldsymbol{x}_0)||p(\boldsymbol{x}_T))}_{L_T} \\ &+ \sum_{t>1} \underbrace{D_{KL}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)||p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))}_{L_{t-1}} \underbrace{-\log p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_1)}_{L_0}\Bigg] \\ &= L_0 + L_1 + \cdots + L_{t-1} + L_T, \end{aligned}$$

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

Treinando um modelo de difusão

- A distribuição posterior da etapa direta é dada por

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

onde

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

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

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

- 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 $\beta_1, \beta_2, \ldots, \beta_T$ são valores constantes e conhecidos, o termo L_T não tem nenhum parâmetro a ser aprendido e, portanto, ele é ignorado durante o treinamento.
- Dado que $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ e $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ são distribuições Normais, a divergência de KL assume uma forma simples e fechada

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

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

- Percebam que 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 $\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$ na definição de $\widetilde{\boldsymbol{\mu}}_t(\boldsymbol{x}_t, \boldsymbol{x}_0)$, podemos reescrever a função L_{t-1} como

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

- A equação acima nos mostra que $\boldsymbol{\mu}_\theta$ (i.e., o modelo) deve predizer $\frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}\right)$ dado $\boldsymbol{x}_t$.
- Dado que $\boldsymbol{x}_t$ está disponível como entrada do modelo, podemos fazer a seguinte parametrização

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

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

Treinando um modelo de difusão

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

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

- Portanto, o modelo é treinado para predizer o ruído adicionado à amostra $\boldsymbol{x}_t$.
- **OBS**.: Os autores de [1] relataram que a qualidade das imagens geradas é melhor ao se descartar λ.
- A rede neural é otimizada usando-se o ***erro quadrático médio*** entre o ruído Gaussiano verdadeiro e o predito.

Treinando um modelo de difusão

- O termo $p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_1)$ de $L_0 = \log p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_1)$ é calculado como

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

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

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

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

O algoritmo de treinamento

Algorithm 1 Training

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

Algoritmo retirado de [1].

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

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

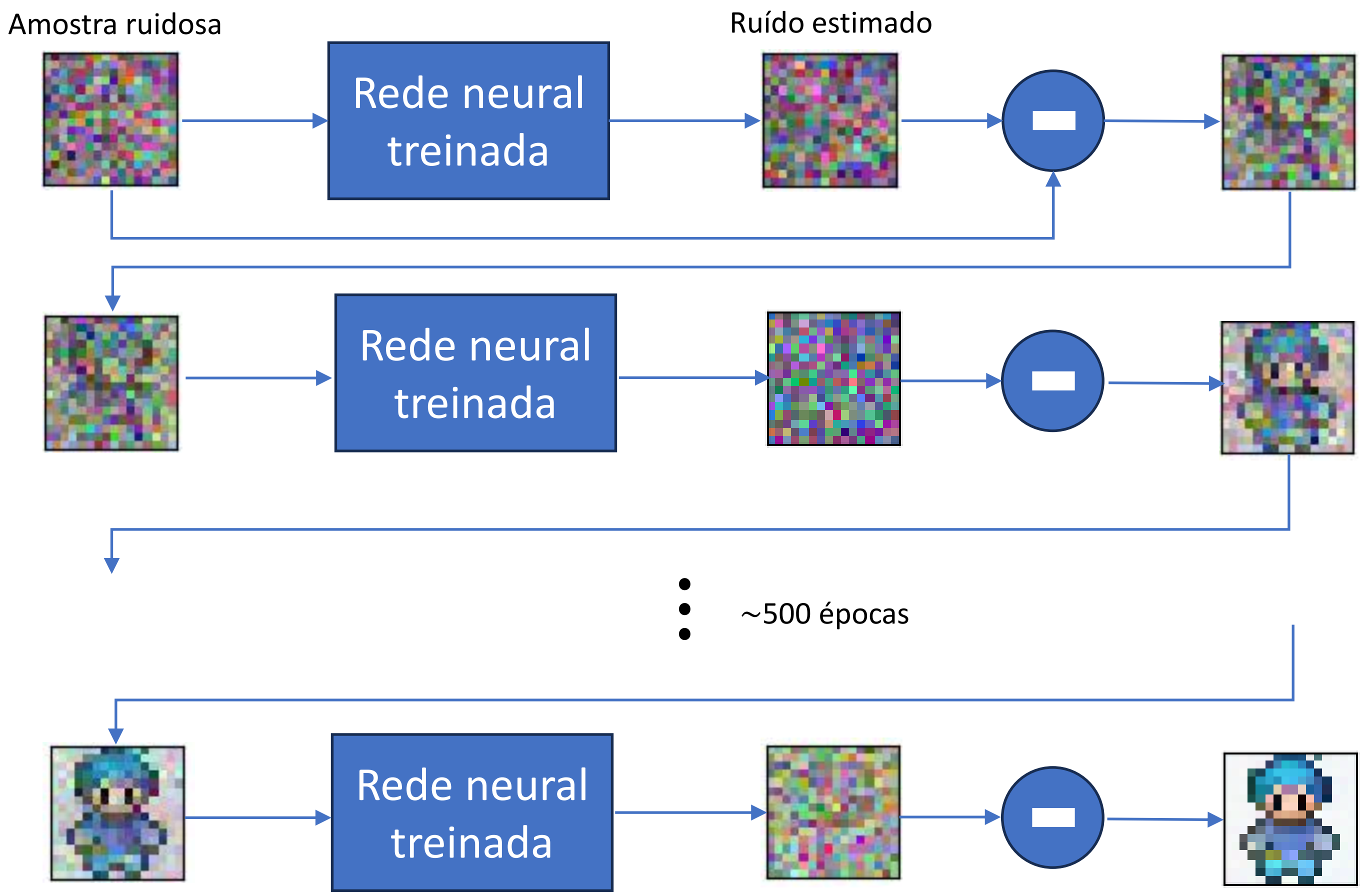
Algorithm 2 Sampling

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

Algoritmo retirado de [1].

- Após o treinamento, para retirar uma amostra da distribuição original, segue-se o algoritmo ao lado.
- Retira-se uma amostra puramente Gaussiana, $\boldsymbol{x}_T$.
- A rede neural estima o ruído adicionado a ela e o remove.
- Antes da nova iteração, ruído Gaussiano é adicionado à $\boldsymbol{x}_{t-1}$.
- Esse processo é repetido até que $t = 1$, nessa iteração, não se adiciona ruído.
- Ao final, o algoritmo retorna a amostra da distribuição original, $\boldsymbol{x}_0$.

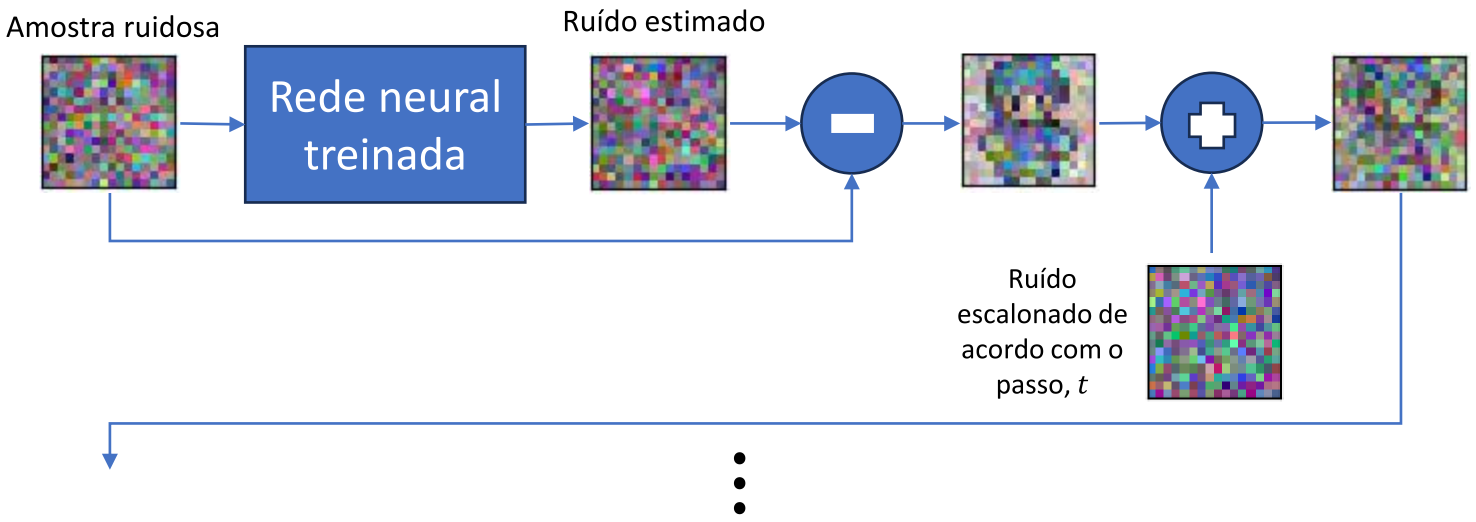
O algoritmo de amostragem (i.e., geração)



OBS.: 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

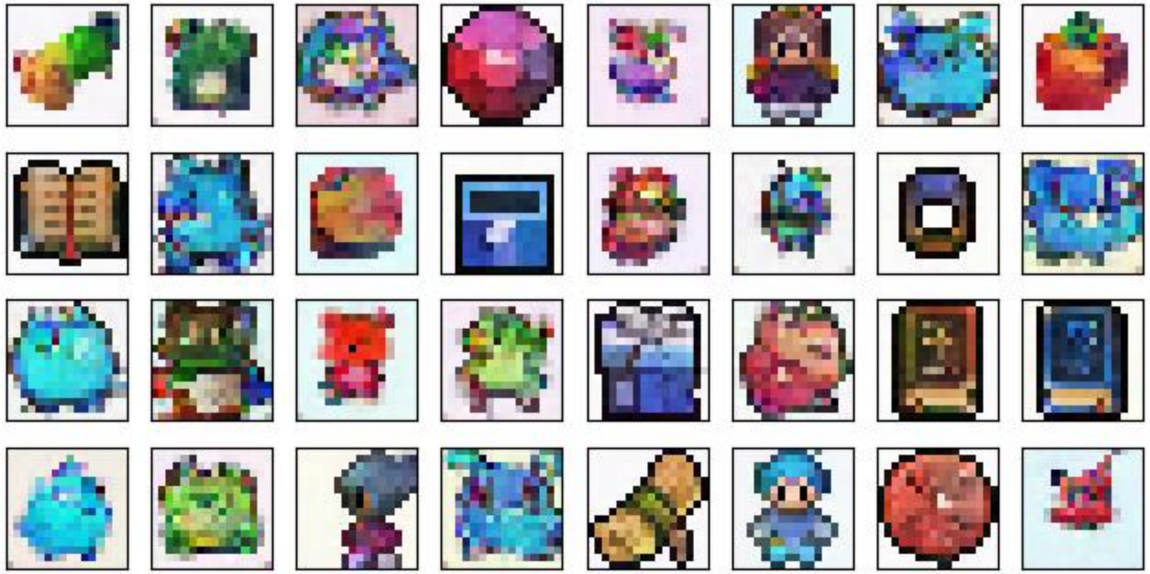
- Em cada passo, a rede tenta predizer o ruído adicionado, até que tenhamos uma imagem.

Detalhes do algoritmo de amostragem

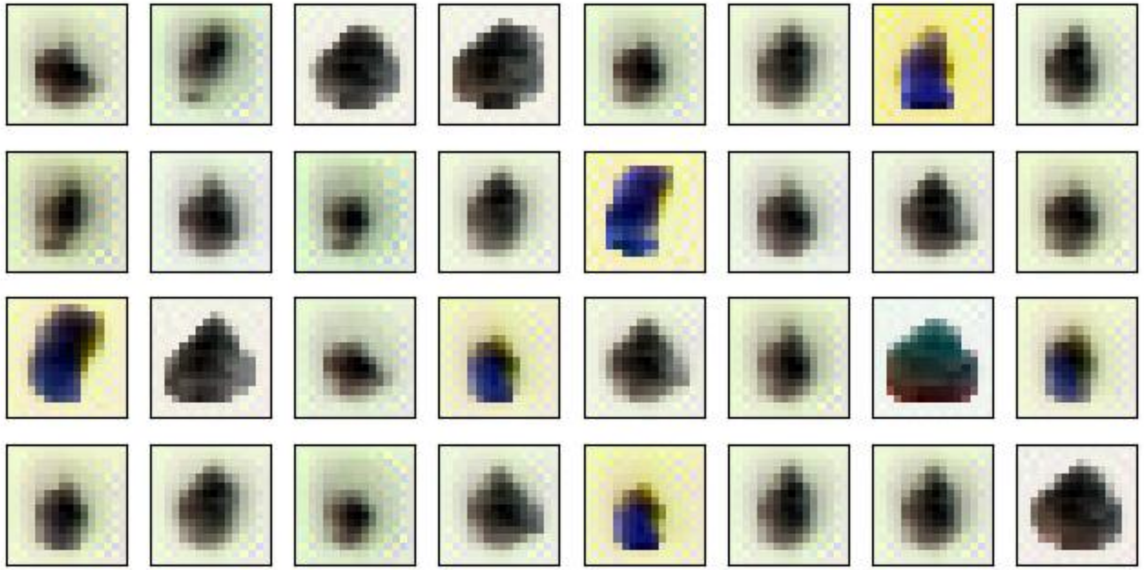


- A rede neural espera ruído Gaussiano normal como entrada.
- Porém, quando removemos o ruído, a amostra deixa de ser distribuída desta forma.
- Portanto, o que devemos fazer após cada passo e antes do próximo é adicionar ruído que é escalonado baseando-se no passo atual, t.

Detalhes do algoritmo de amostragem



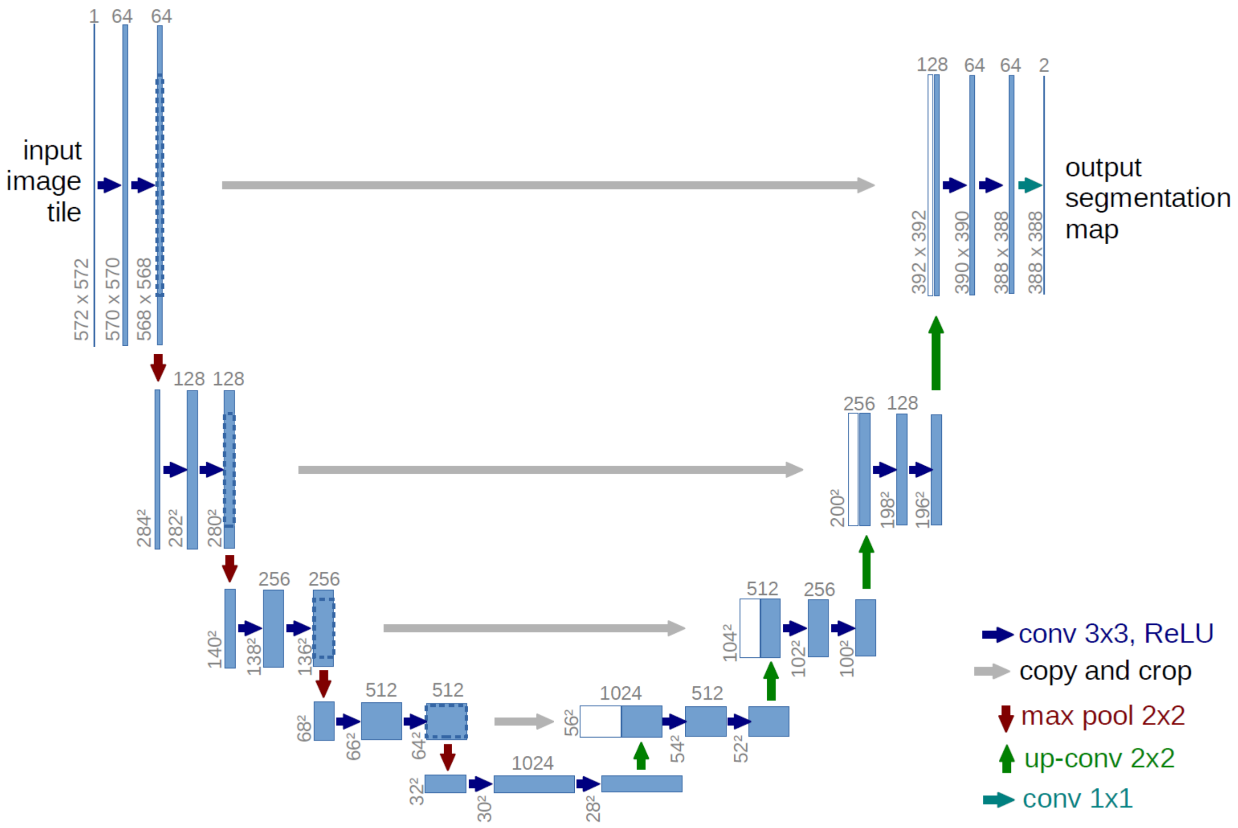
Com adição de ruído antes do próximo passo.



Sem adição de ruído antes do próximo passo.

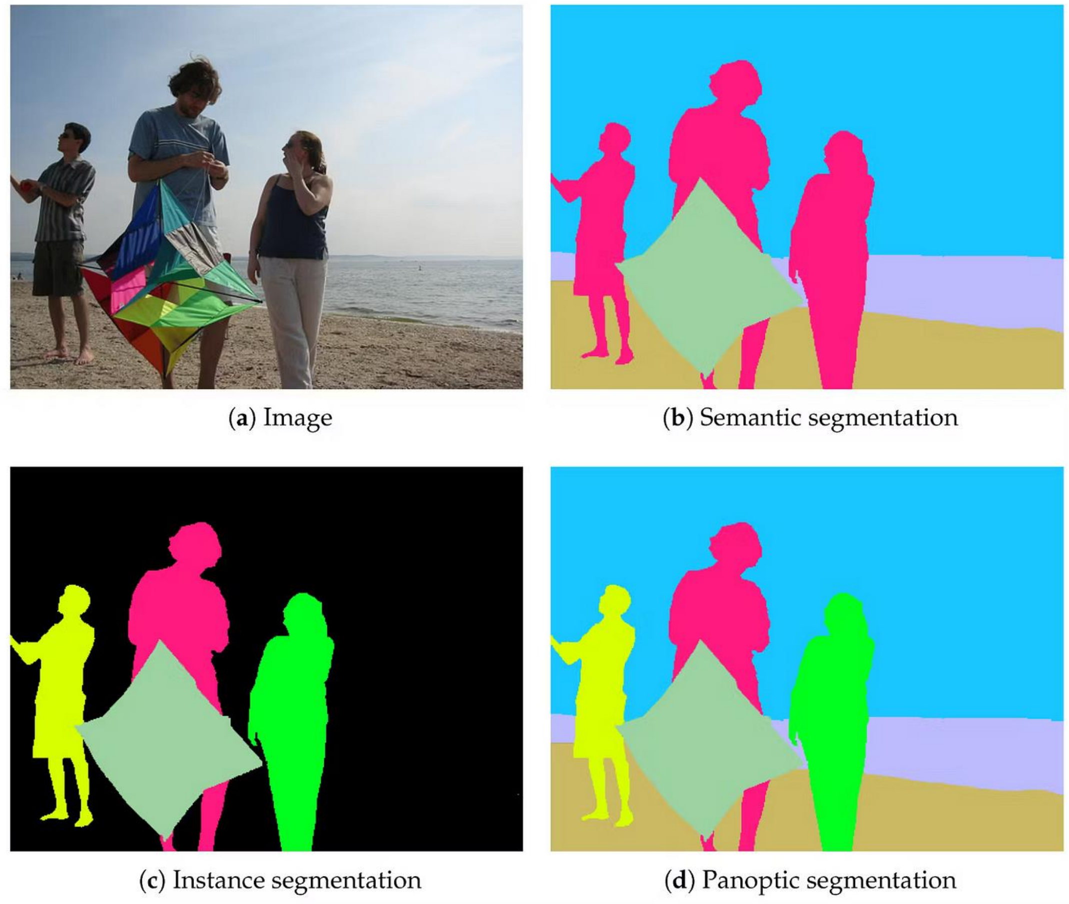
- Empiricamente, a adição de ruído ajuda a estabilizar a rede neural para que ela não entre em colapso e gere amostras próximas da média do conjunto de dados.

U-Net



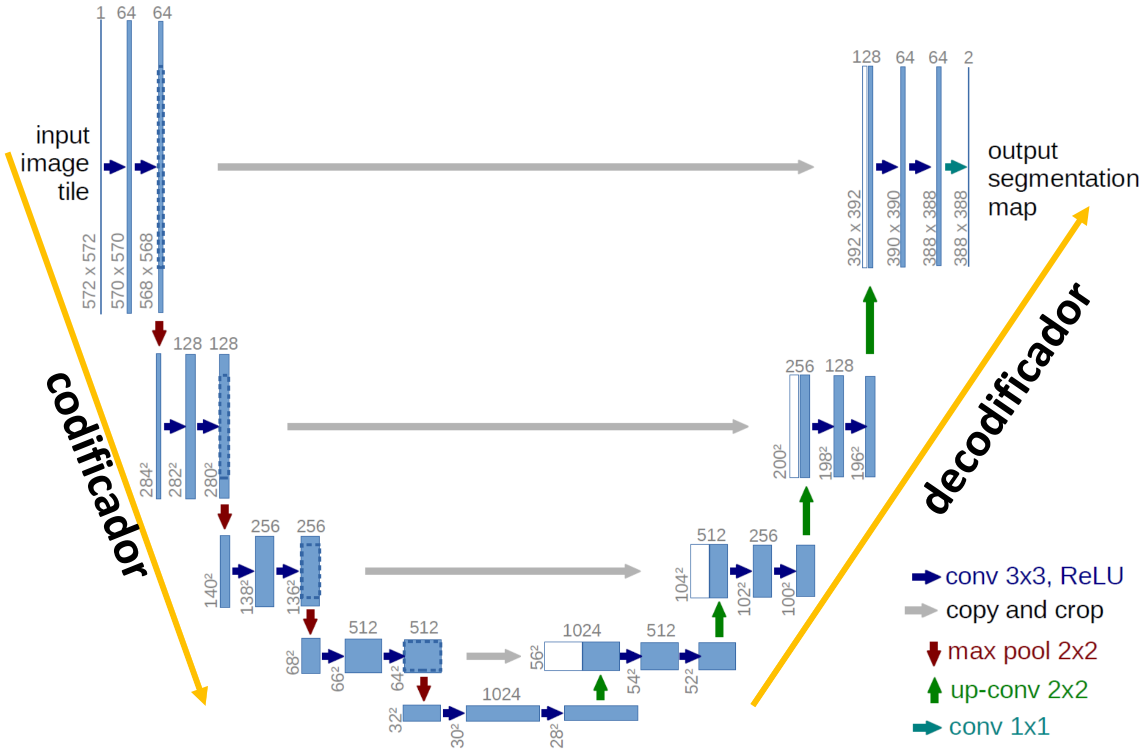
- A rede neural recebe uma amostra (e.g., imagem) com ruído em um determinado passo, t, e retorna uma predição do ruído.
- Observe que o ***ruído predito deve ter a mesma dimensão da amostra de entrada***.
- Assim, os autores de [1] optaram por usar uma U-Net, introduzida por Ronneberger et al. em [2] para ***segmentação de imagens*** médicas.
  - U-Net é uma rede neural convolucional.

U-Net



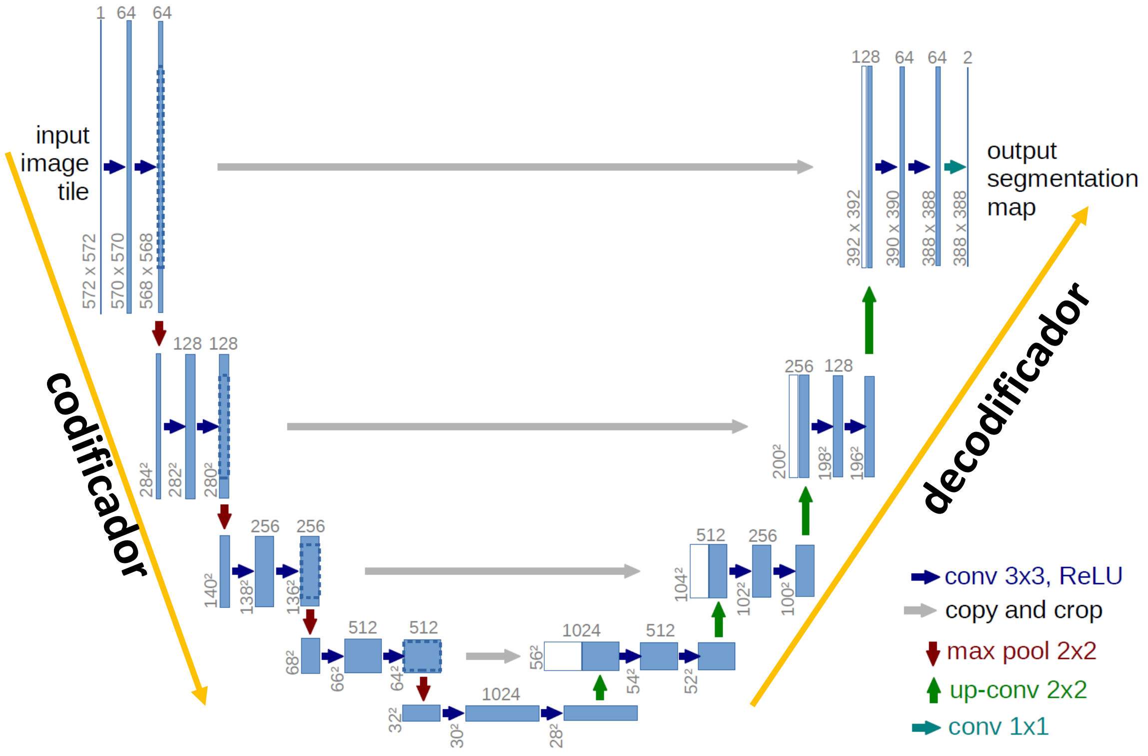
- A segmentação de imagens é uma técnica de visão computacional que particiona uma imagem em grupos de pixels (i.e., segmentos de imagem) para detecção de objetos.
  - Ou seja, classifica os pixels em classes.
- Segmentação semântica: detecta pixels da mesma ***classe*** (e.g., pessoas).
- Segmentação por instância: detecta pixels de ***diferentes instâncias de uma classe*** (e.g., diferentes pessoas).
- Segmentação panóptica: combina as duas anteriores.

U-Net



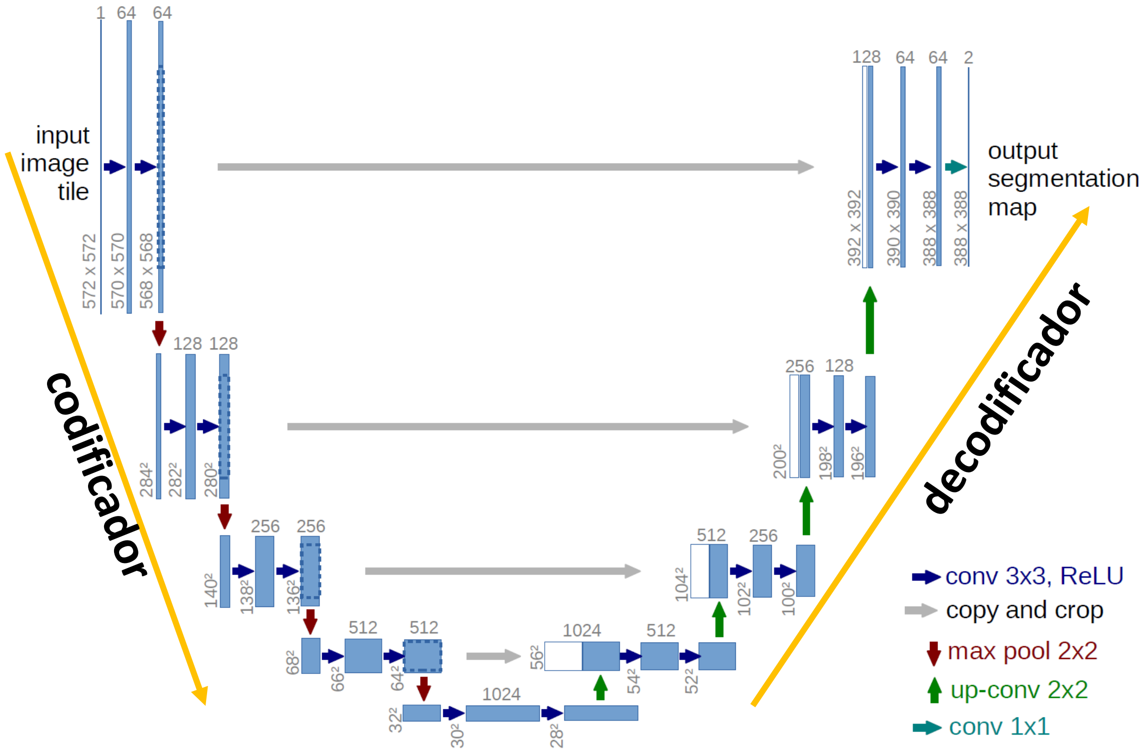
- Uma U-Net possui uma camada de "***gargalo***" no meio de sua arquitetura entre o codificador e o decodificador.
- Esse gargalo garante que a rede aprenda apenas as informações mais importantes.
- O gargalo encontra informações latentes, i.e., ocultas, nos dados de entrada.

U-Net



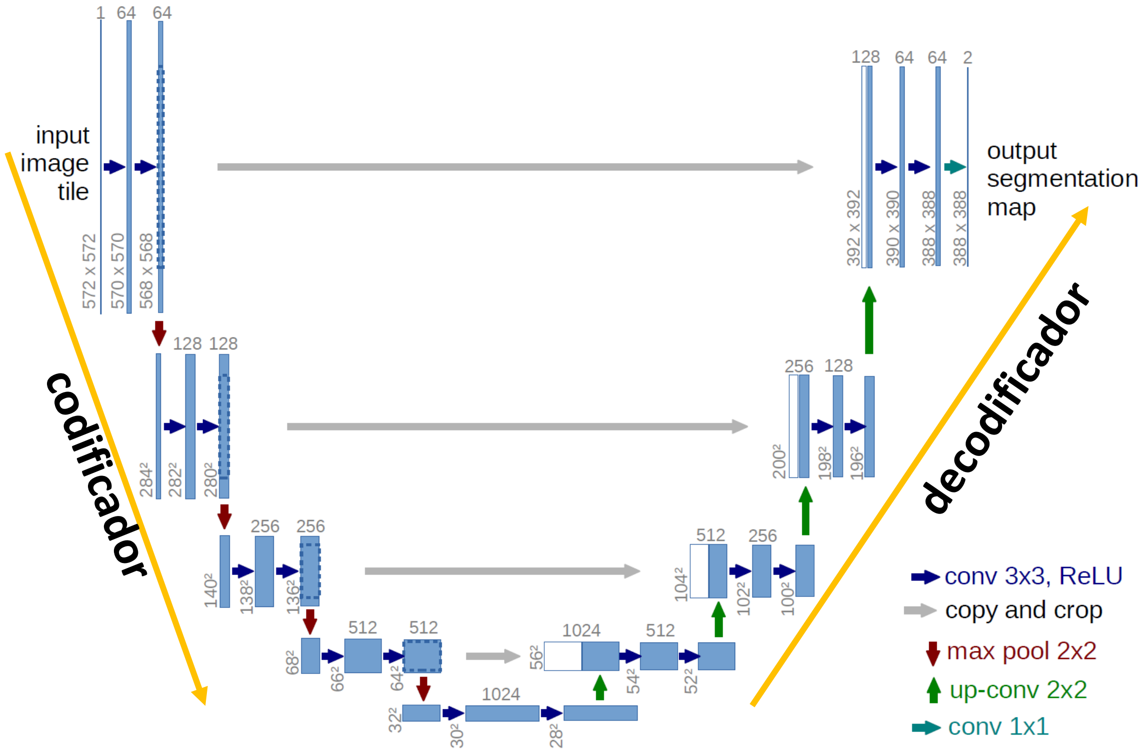
- O codificador primeiro codifica uma imagem em uma representação oculta de menor resolução.
  - Ou seja, torna a entrada ***menor*** em termos de ***resolução espacial***.
- Em seguida, o decodificador decodifica essa representação oculta de volta em uma imagem com a mesma resolução inicial.
  - Ou seja, ***aumenta a resolução espacial***.

U-Net: Codificador



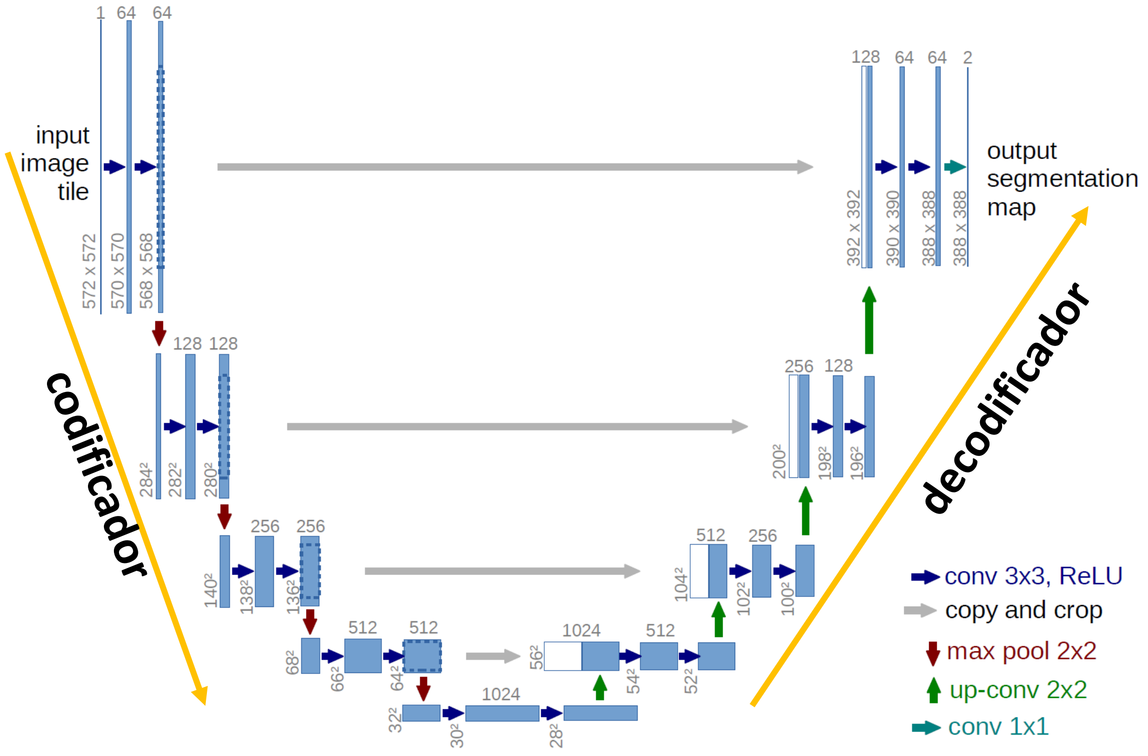
- O ***codificador*** é responsável por identificar as ***características*** relevantes na imagem de entrada.
- Suas camadas realizam ***convoluções*** (seguidas por ativação ReLU e *max pooling*) que reduzem a resolução espacial dos ***mapas de características***, capturando assim representações cada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

U-Net: Codificador



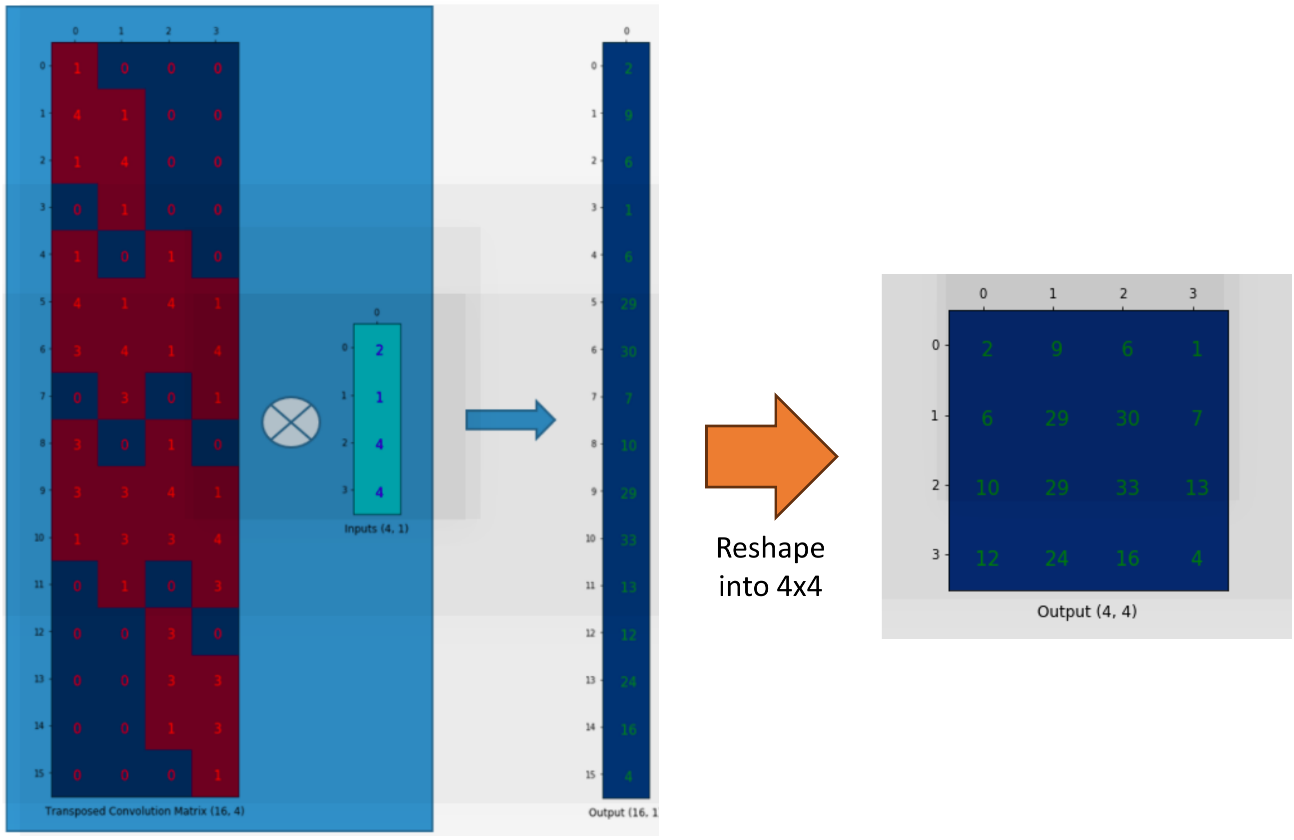
- Durante a ***codificação***, a ***informação espacial é reduzida*** enquanto a ***informação das características é aumentada*** devido ao aumento do número de canais.
- A tarefa do codificador é semelhante à de outras CNNs.

U-Net: Decodificador



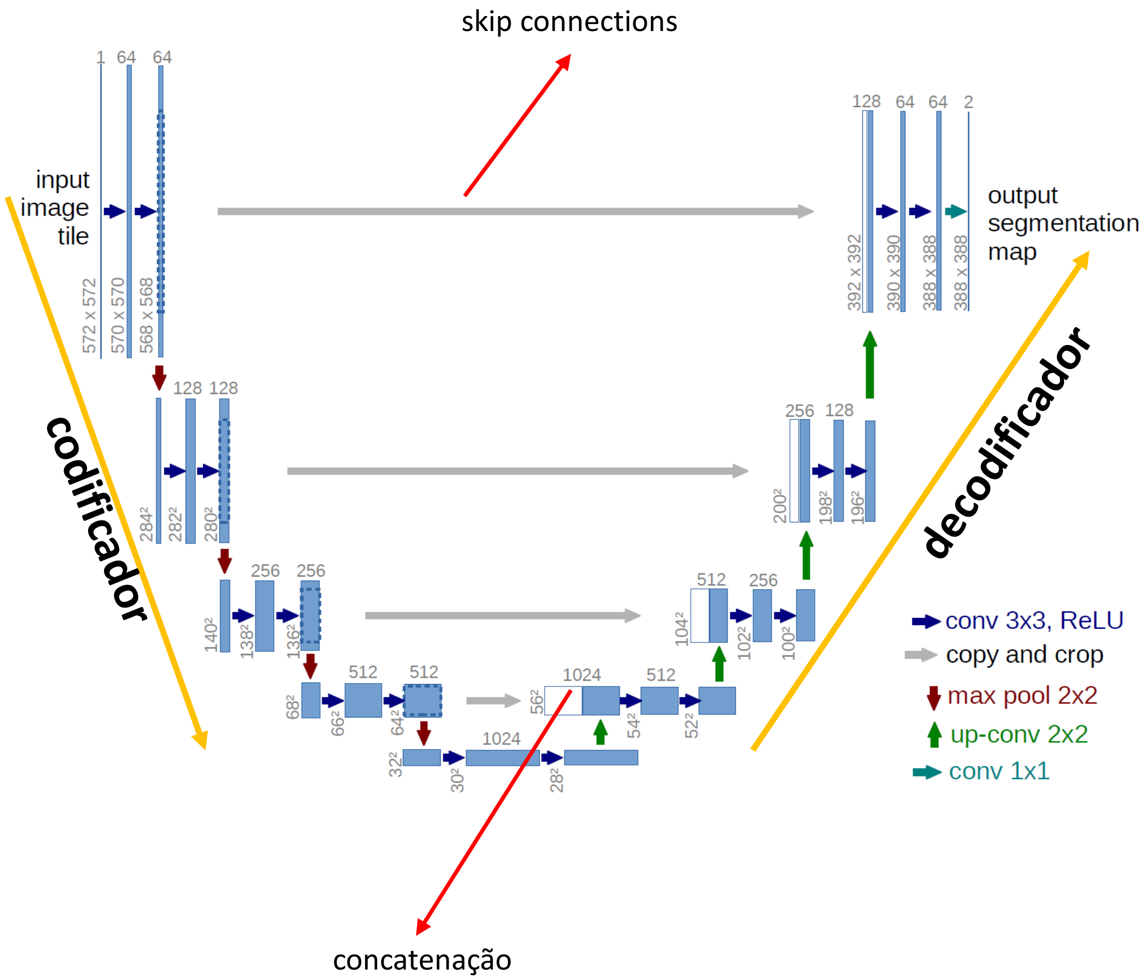
- O ***decodificador*** é responsável por decodificar as informações abstratas (i.e., latentes) capturadas pelo codificador, mantendo a resolução espacial da entrada.
- Suas camadas aumentam a resolução dos mapas de características através de ***convoluções transpostas***.

U-Net: Decodificador



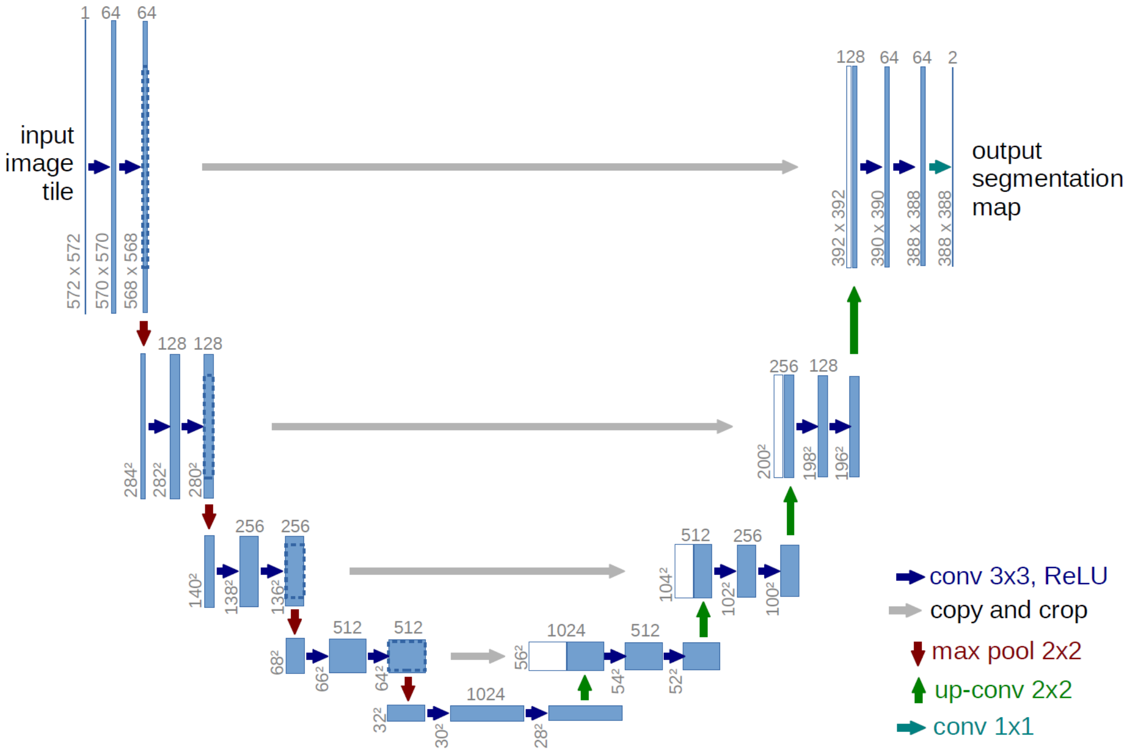
- Nas camadas de ***convolução transposta***, as operações de *pooling* são substituídas por operações de ***upsampling***.
  - Ou seja, essas camadas aumentam a resolução da saída.
- A operação de convolução transposta é similar a de convolução, sendo a única diferença é que aqui multiplicamos a matriz do *kernel* transposta pela entrada achatada e depois realizamos um *reshape*.

U-Net: Decodificador



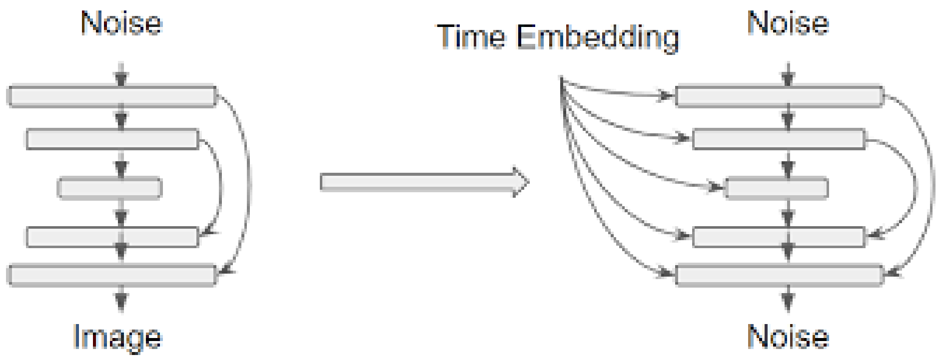
- Em geral, as CNNs, incluindo o codificador, tendem a esquecer certas características ao longo das camadas convolucionais.
- As ***skip connections*** ajudam a ***preservar*** as ***informações espaciais perdidas na codificação***.
  - Isso ajuda as camadas decodificadoras a localizar as características com mais precisão.
- Cada nível do decodificador concatena os mapas de características locais com características de alta resolução do caminho do codificador.

U-Net: Informação temporal



- Até o momento, vimos que a U-Net recebe um tensor com determinadas dimensões como entrada e gera um tensor com as mesmas dimensões em sua saída.
  - Ou seja, ela recebe uma imagem e gera um tensor com a predição do ruído adicionado àquela imagem.

U-Net: Informação temporal



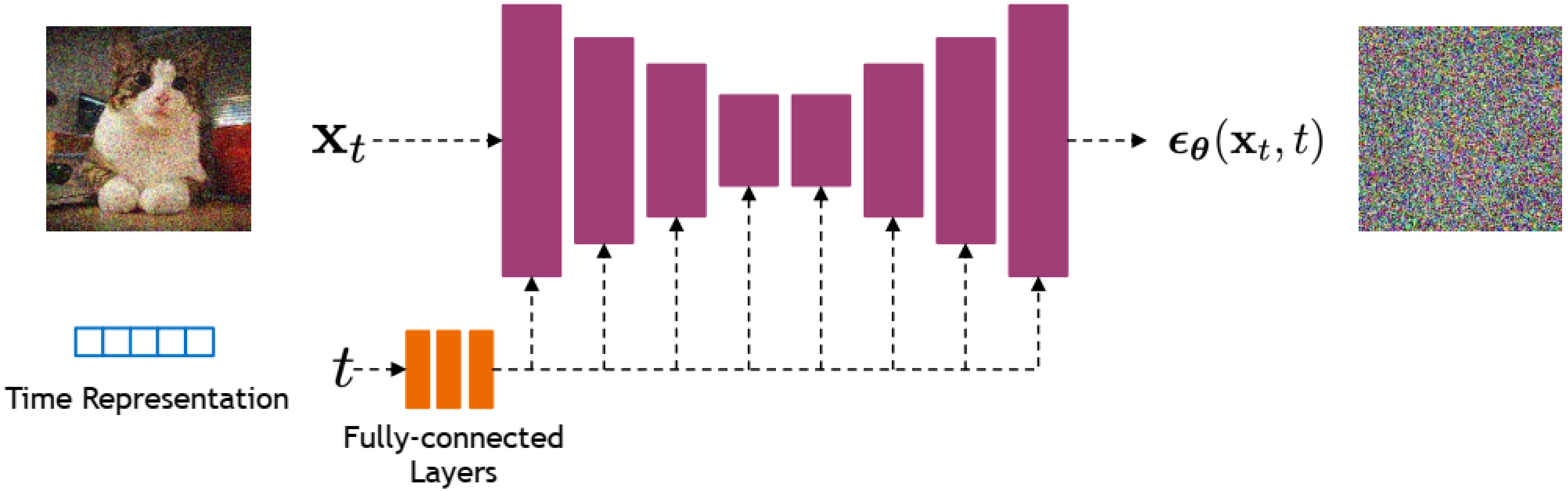
- Porém, lembrem-se que a rede deve receber também como entrada o passo, t (nível de ruído)

$$\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t).$$

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

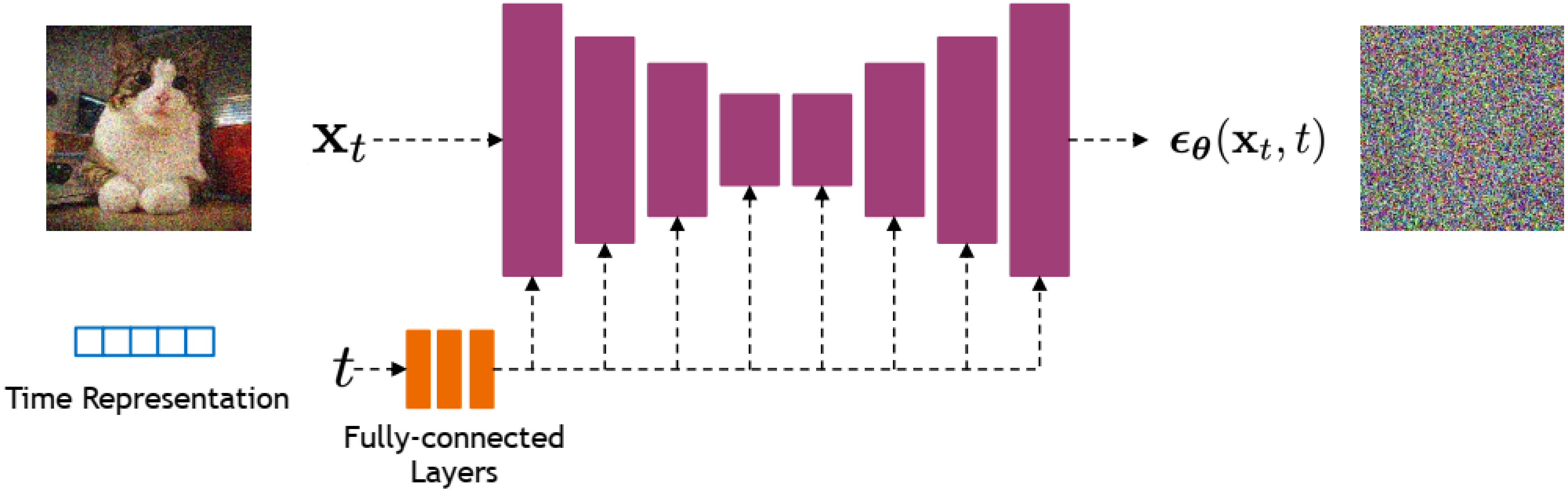
Positional embeddings

- Em geral, o passo de tempo t é ***adicionado*** 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 para cada imagem e consiga predizer 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

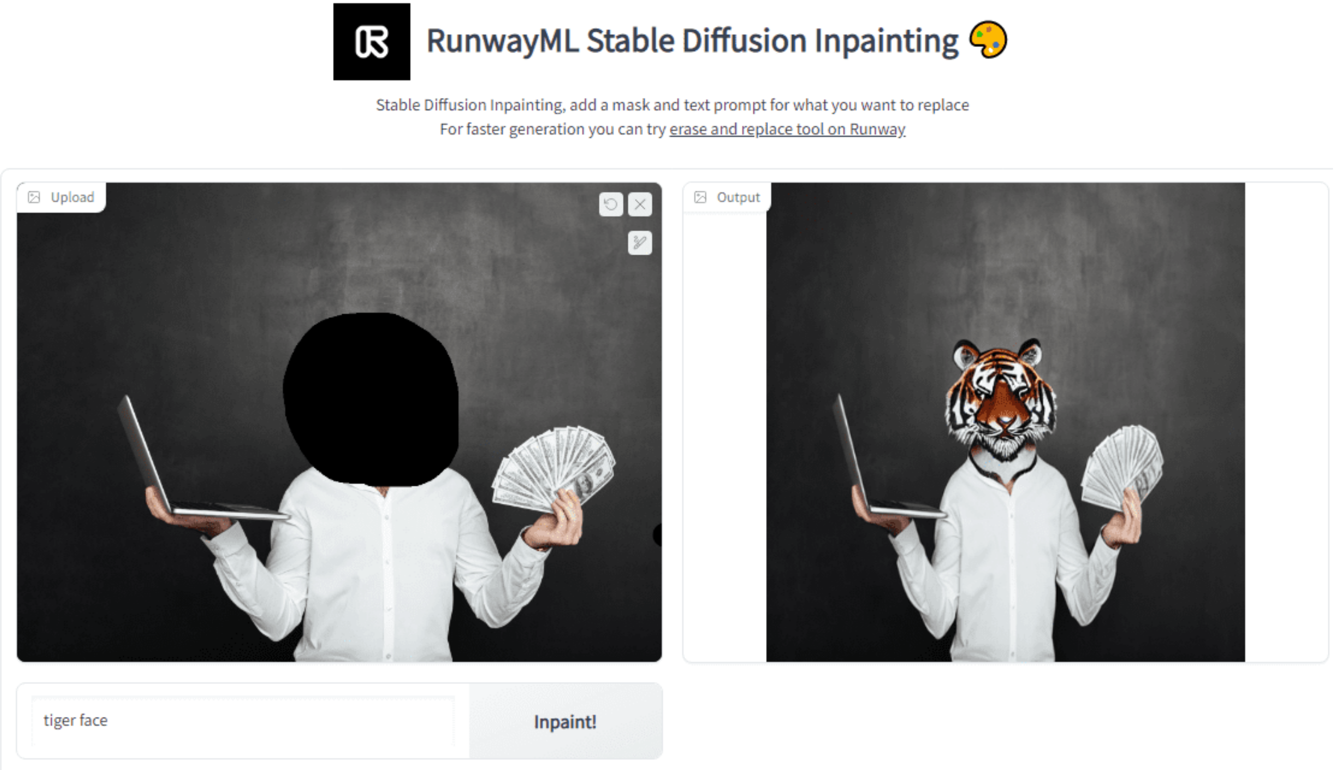
- Geração de vídeos a partir de *prompts* de texto.
- Para isso, o modelo precisa ser modificado para receber como entrada o texto, através de ***word embeddings***.
- Exemplo: MagicVideo.

Aplicações

Picture of Chef Chopping Carrots on a cutting board

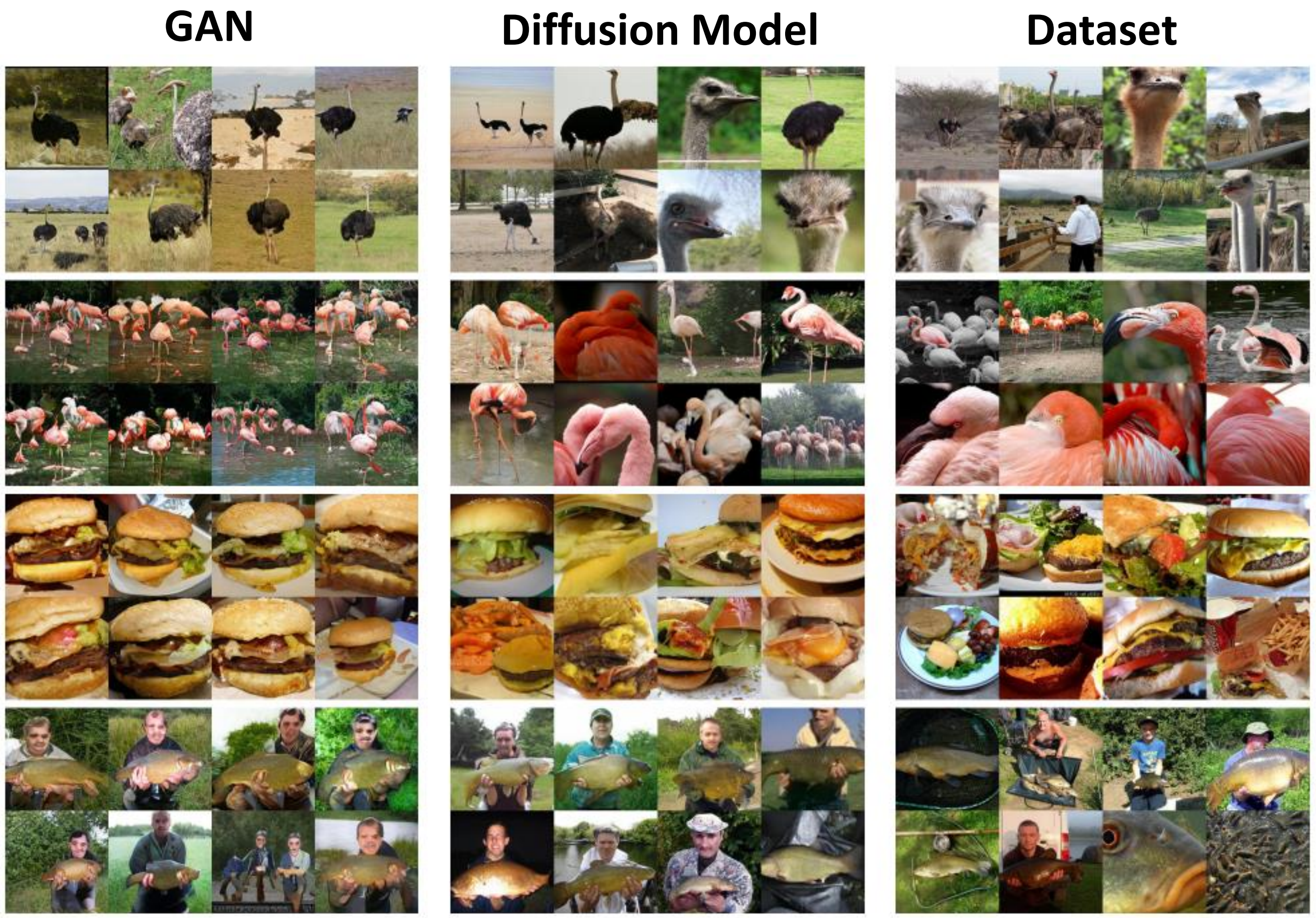
- Geração de imagens a partir de *prompts* de texto.
- Exemplos: Midjourney, Google Imagen, and DALL-E

Aplicações



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

Comparação com outros algoritmos



- O modelo de difusão gera mais modos (i.e., diversidade) do que a GAN, como, por exemplo, cabeças de avestruz ampliadas, flamingos únicos, diferentes orientações de cheeseburgers e um peixe sem nenhum humano segurando-o [4].

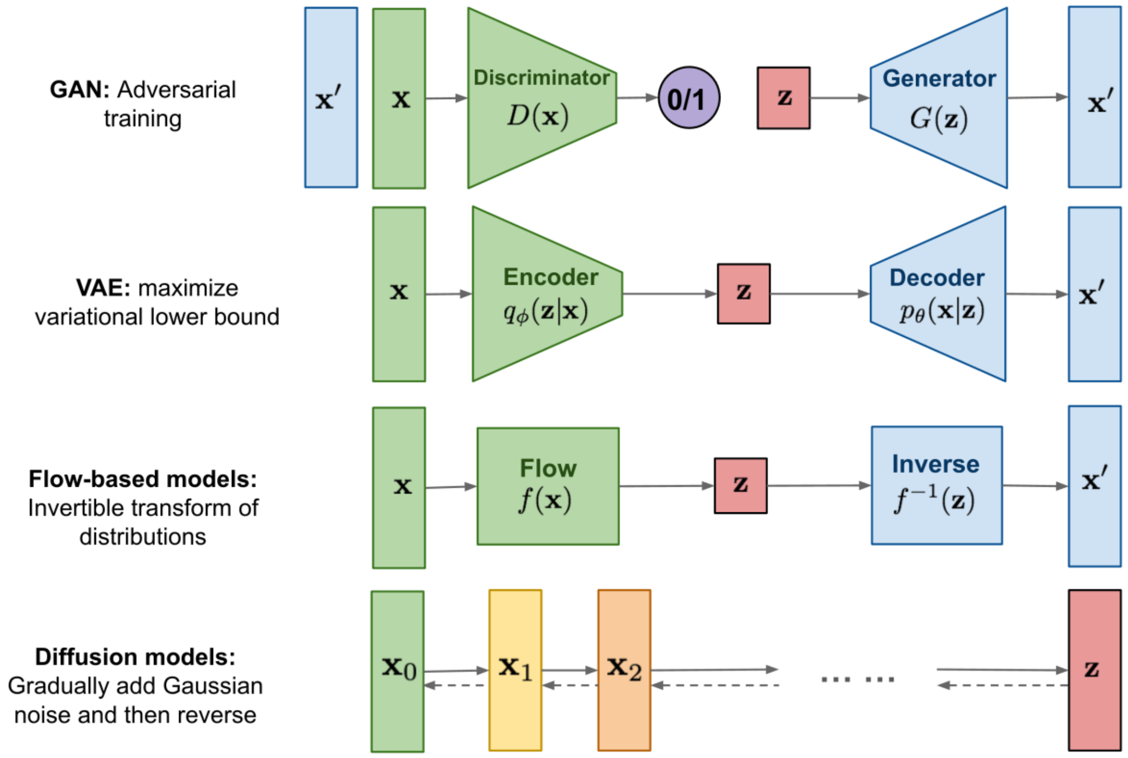
Comparação com outros algoritmos

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**

Model	FID	sFID	Prec	Rec
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:
  - Generative Adversarial Networks (GANs).
  - Variational Autoencoders (VAEs).
  - Flow models.

Exemplos

- [Sampling]
- [Training]

python™ + TensorFlow + jupyter + Google colab

Quiz

- Modelos de difusão

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!