

# Hand-on Deep Learning - Aula 5

## Modelos Generativos

Camila Laranjeira<sup>12</sup>, Hugo Oliveira<sup>12</sup>, Pedro H. Barros<sup>13</sup>

<sup>1</sup>Programa de Pós-Graduação em Ciência da Computação (PPGCC)  
Universidade Federal de Minas Gerais

<sup>2</sup>Interest Group in Pattern Recognition and Earth Observation (PATREO)  
Universidade Federal de Minas Gerais

<sup>3</sup>Wireless Informational Sensing Embedded systems Modeling Algorithms and Protocols (WISEMAP)  
Universidade Federal de Minas Gerais

07 de Março, 2020





# Agenda

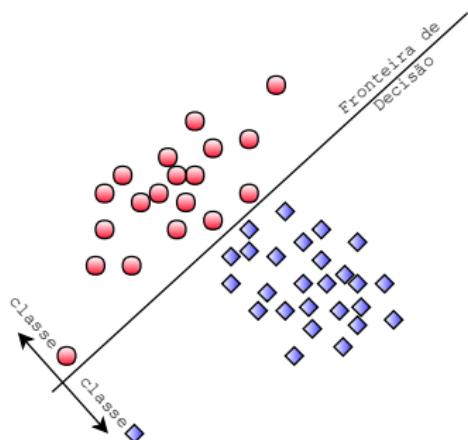
- 1 Introdução**
- 2 Modelos Generativos**
  - Taxonomia de Modelos Generativos
- 3 PixelRNN and PixelCNN**
- 4 AutoEncoders**
  - Variational AutoEncoder (VAE)
- 5 Generative Adversarial Networks**
  - Histórico das GANs

# Agenda

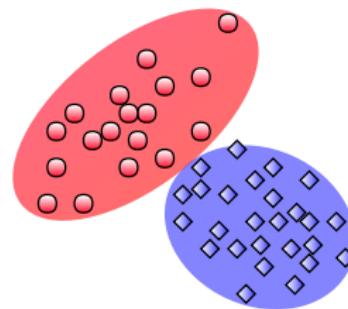


- 1** **Introdução**
- 2** **Modelos Generativos**
  - Taxonomia de Modelos Generativos
- 3** **PixelRNN and PixelCNN**
- 4** **AutoEncoders**
  - Variational AutoEncoder (VAE)
- 5** **Generative Adversarial Networks**
  - Histórico das GANs

# Modelos Generativos vs. Modelos Discriminativos



**(a)** Modelo Discriminativo

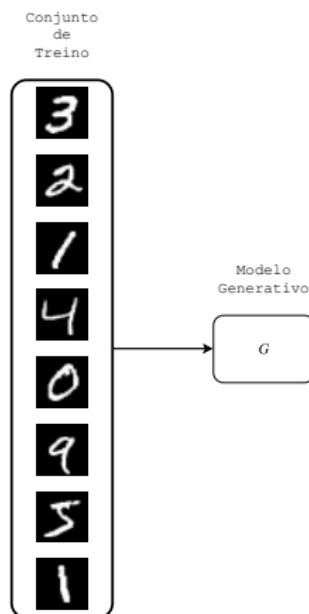


**(b)** Modelo Generativo

**Figura:** Classificação de Modelos em Machine Learning.



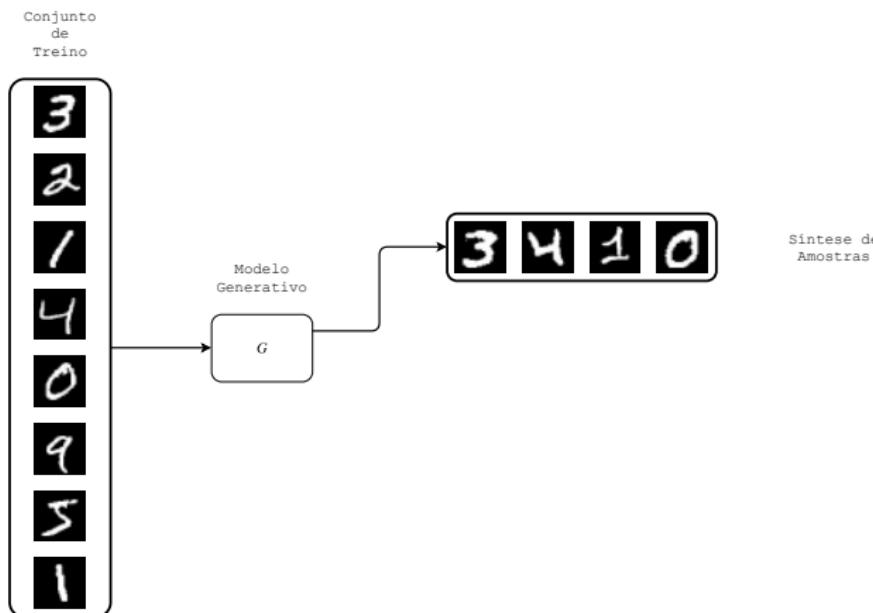
# Modelos Generativos



**Figura:** O que fazer com uma modelagem generativa?



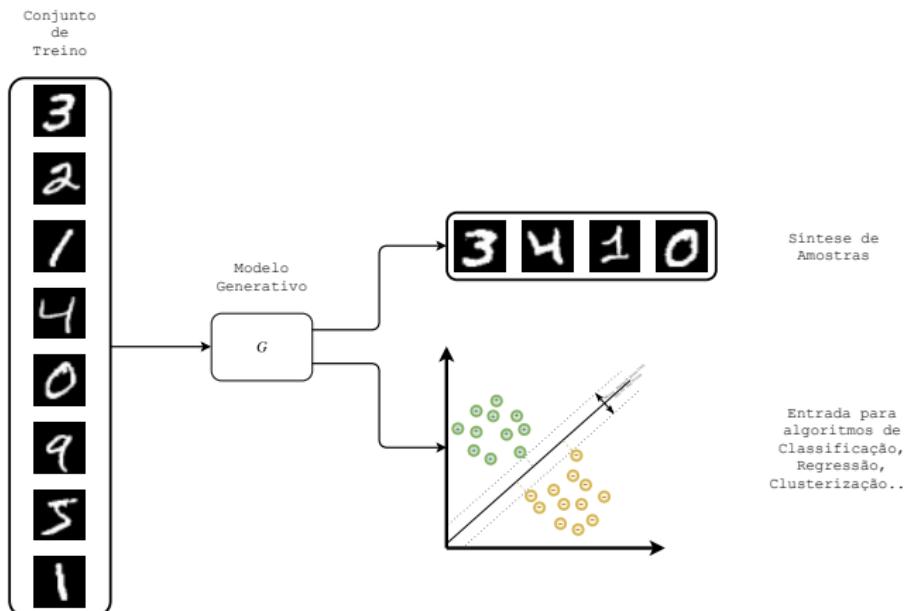
# Modelos Generativos



**Figura:** O que fazer com uma modelagem gerativa?



# Modelos Generativos



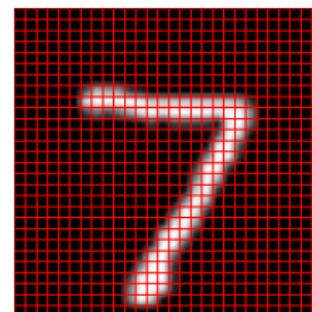
**Figura:** O que fazer com uma modelagem generativa?

# Modelos Generativos



## Por que estudar modelos generativos? [1]

É uma ótima forma de testar a habilidade de modelar distribuições de probabilidade com um **alto número de dimensões**.



# Modelos Generativos



## Por que estudar modelos generativos? [1]

Modelos Generativos podem ser incorporados em algoritmos de **Reinforcement Learning**<sup>1</sup>. Um Modelo Generativo treinado para **planejamento** pode aprender uma distribuição condicional sobre estados futuros do mundo, dado o estado atual e ações hipotéticas do agente [2, 3].

<sup>1</sup> <https://skymind.ai/wiki/deep-reinforcement-learning>

# Modelos Generativos



## Por que estudar modelos generativos? [1]

Modelos Generativos podem ser treinados com dados incompletos. Muito usados para **Aprendizado Semi-Supervisionado**.

# Modelos Generativos



## Por que estudar modelos generativos? [1]

Modelos Generativos permitem a geração de saídas multimodais [4]. Há tarefas que possuem várias respostas corretas para uma única entrada (i.e. geração de amostras sintéticas, tradução de imagens etc).

# Modelos Generativos



## Por que estudar modelos generativos? [1]

Modelagem de **tarefas que não possuem uma função de perda bem caracterizada** (i.e. Unsupervised Image-to-Image Translation, geração de imagens fotorrealistas de um certo domínio, extração não-supervisionada de características etc.).

# Modelos Generativos



## Por que estudar modelos generativos? [1]

Várias tarefas requerem **geração de amostras realísticas** de uma distribuição (i.e. modelos baseados em MCMC, Data Augmentation, Transfer Learning etc).



# Modelos Generativos

## Tarefas de Geração de Amostras [1]

Superresolução em imagens. Há vários outputs possíveis para uma única imagem de baixa resolução. O modelo deve selecionar uma imagem que seja uma amostra da distribuição de probabilidades de várias outras imagens que foram usadas na etapa de treino. A escolha de uma imagem que seja a média de todas as possíveis imagens resulta em imagens borradass e não-realísticas.



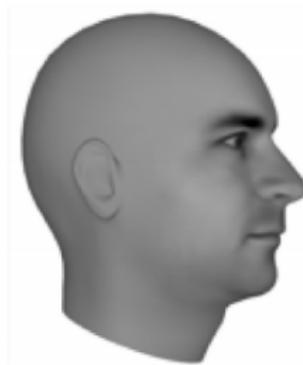
**Figura:** Superresolução em imagens usando Modelos Generativos.



# Modelos Generativos

## Tarefas de Geração de Amostras [1]

Predição de frames em vídeo. A escolha de uma imagem que seja a média de todas as possíveis imagens resulta em imagens borradass e não-realísticas.



(a) Ground Truth



(b) MSE



(c) Modelo Generativo

**Figura:** Predição de frames em vídeo [5].



# Modelos Generativos

## Tarefas de Geração de Amostras [1]

Tradução de Imagens. Muitas vezes o domínio target é muito mais “aberto” que o domínio source, logo, há a possibilidade de outputs multimodais. **A escolha de uma imagem que seja a média de todas as possíveis imagens resulta em imagens borradass e não-realísticas.**



**Figura:** Tradução de Imagens [6].

# Modelos Generativos



## Tarefas de Geração de Amostras [1]

Criação de imagens “artísticas”. Muitas abordagens recentes usando modelos generativos conseguem fazer **transferência de estilo em imagens** [4]. Outras abordagens são baseadas na criação ou edição interativa de imagens [7, 8].



**Figura:** Exemplos de transferência de estilo<sup>2</sup>.

<sup>2</sup><https://news.softpedia.com/news/adobe-s-deep-photo-style-transfer-tech-is-and-works-exactly-how-it-sounds-514425.shtml>



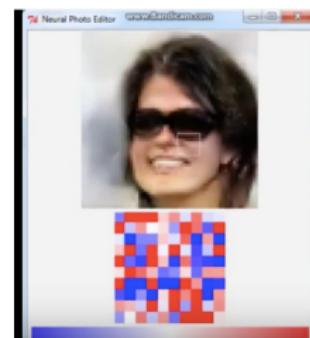
# Modelos Generativos

## Tarefas de Geração de Amostras [1]

Criação de imagens “artísticas”. Muitas abordagens recentes usando modelos generativos conseguem fazer transferência de estilo em imagens [4]. Outras abordagens são baseadas na **criação ou edição interativa de imagens** [7, 8].



(a) [7]



(b) [8]

**Figura:** Edição interativa de imagens usando Modelos Generativos.



# Agenda

## 1 Introdução

## 2 Modelos Generativos

- Taxonomia de Modelos Generativos

## 3 PixelRNN and PixelCNN

## 4 AutoEncoders

- Variational AutoEncoder (VAE)

## 5 Generative Adversarial Networks

- Histórico das GANs

# Maximum Likelihood Estimation



- Estimativa de Máxima Verossimilhança (MLE) é a base para modelagens generativas
  - Ronald Fisher
- Servem de base também para inferência bayesiana
- Combina evidências (dados) com distribuições *a priori*
- Modelo de estimação *não-enviesado* com a menor variância possível

# Maximum Likelihood Estimation



## Maximum Likelihood Estimation

Nem todo modelo gerativo é baseado em MLE (i.e. GANs), mas entender esse método de estimação é imprescindível para compreender a área de Modelos Gerativos.



# Maximum Likelihood Estimation

- Modelo estatístico:  $p_{model}$
- Parâmetros do modelo:  $\theta$
- Amostras de treino:  $x^{(i)}, i = 1, 2, \dots, m$
- Modelagem:  $\prod_{i=1}^m p_{model}(x^{(i)}; \theta)$

MLE

$$\theta^* = \arg \max_{\theta} \prod_{i=1}^m p_{model}(x^{(i)}; \theta)$$



# Maximum Likelihood Estimation

- Modelo estatístico:  $p_{model}$
- Parâmetros do modelo:  $\theta$
- Amostras de treino:  $x^{(i)}, i = 1, 2, \dots, m$
- Modelagem:  $\prod_{i=1}^m p_{model}(x^{(i)}; \theta)$

## MLE

$$\theta^* = \arg \max_{\theta} \prod_{i=1}^m p_{model}(x^{(i)}; \theta)$$



# Maximum Likelihood Estimation

- Modelo estatístico:  $p_{model}$
- Parâmetros do modelo:  $\theta$
- Amostras de treino:  $x^{(i)}, i = 1, 2, \dots, m$
- Modelagem:  $\prod_{i=1}^m p_{model}(x^{(i)}; \theta)$

## MLE

$$\theta^* = \arg \max_{\theta} \sum_{i=1}^m \log p_{model}(x^{(i)}; \theta)$$



# Maximum Likelihood Estimation

## MLE as KL Divergence Optimizer

É possível entender MLE como uma minimização da **Divergência de Kullback-Leibler** entre a distribuição  $p_{data}$  dos dados e a distribuição  $p_{model}$  do modelo.



# Maximum Likelihood Estimation

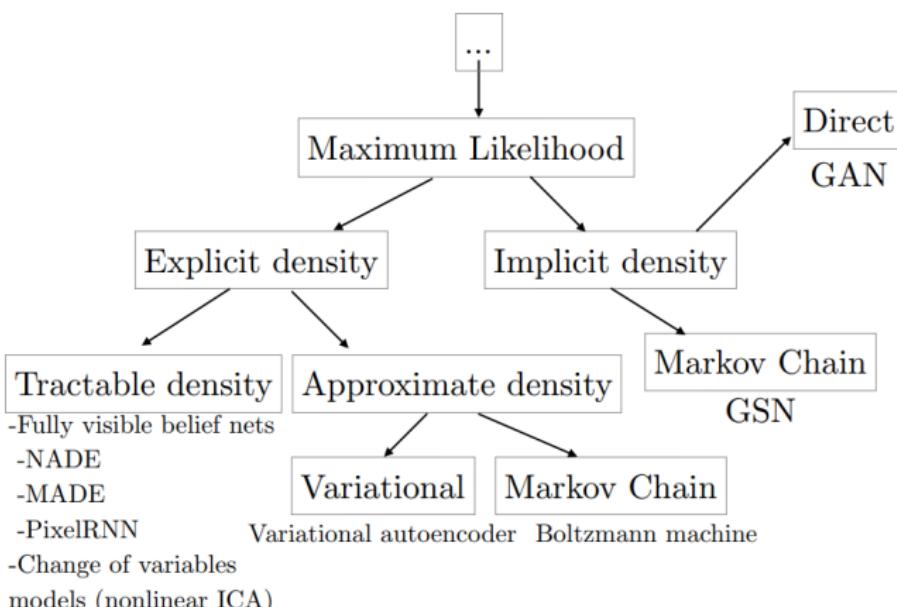
- Problemas:
  - A modelagem requer uma representação explícita de  $p_{model}$
  - Funciona apenas para funções “comportadas” (i.e. deriváveis, contínuas...)
  - Difícil (se não impossível) de calcular para distribuições de probabilidade multimodais
  
- Como estimar essas distribuições mais difíceis de serem modeladas?



# Maximum Likelihood Estimation

- Problemas:
  - A modelagem requer uma representação explícita de  $p_{model}$
  - Funciona apenas para funções “comportadas” (i.e. deriváveis, contínuas...)
  - Difícil (se não impossível) de calcular para distribuições de probabilidade multimodais
- Como estimar essas distribuições mais difíceis de serem modeladas?

# Taxonomia de Modelos Generativos



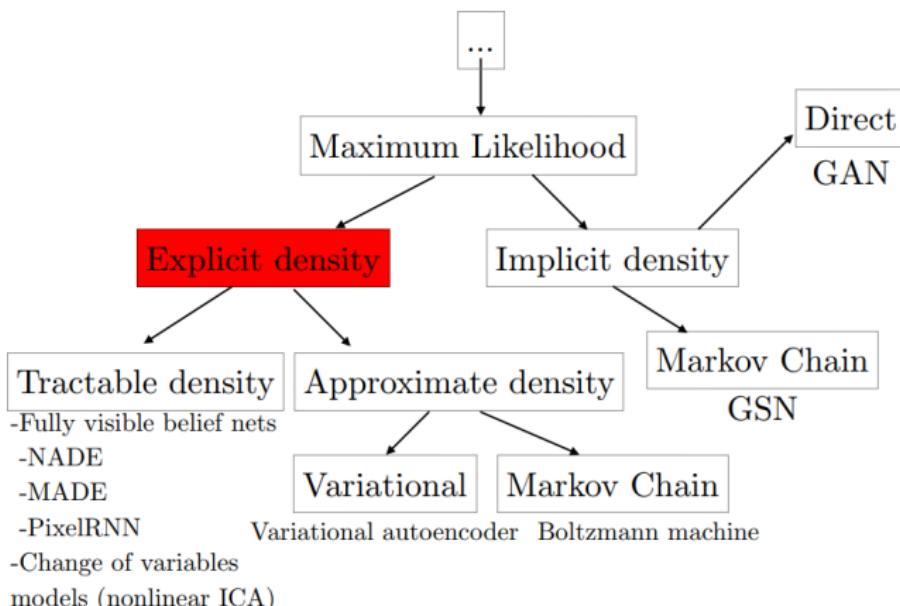
**Figura:** Taxonomia dos modelos generativos [1].

# Agenda



- 1** Introdução
- 2** Modelos Generativos
  - Taxonomia de Modelos Generativos
- 3** PixelRNN and PixelCNN
- 4** AutoEncoders
  - Variational AutoEncoder (VAE)
- 5** Generative Adversarial Networks
  - Histórico das GANs

# Taxonomia de Modelos Generativos



**Figura:** Taxonomia dos modelos generativos [1].



# Modelos de Densidade Explícita

- Geram uma função de densidade explícita  $p_{model}(x; \theta)$
- O processo de optimização é simples: basta plugar a equação que define a distribuição na expressão da verossimilhança e seguir o gradiente

## Dificuldades Associadas

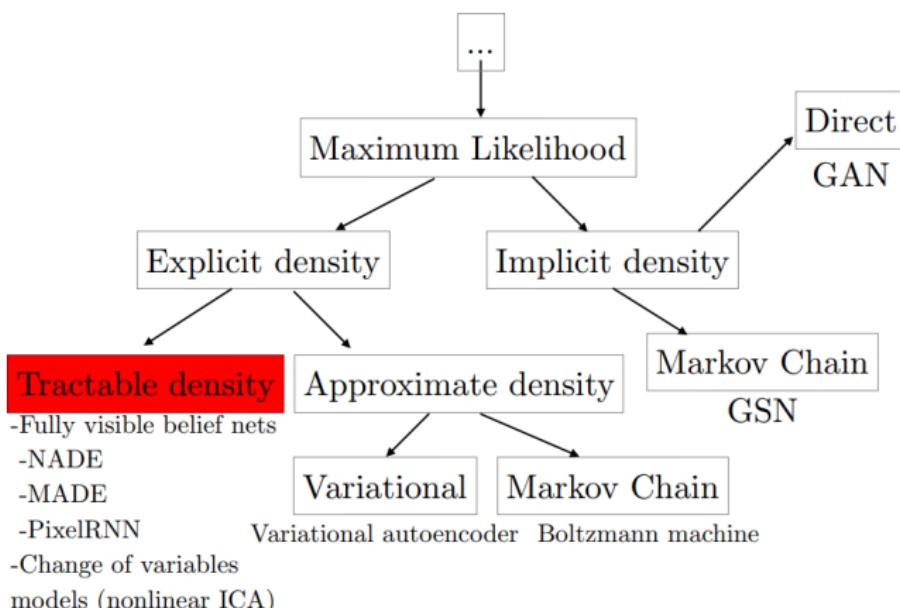
A maior dificuldade é em criar modelos que capturem toda a complexidade dos dados que estão sendo modelados deixando os modelos tratáveis computacionalmente

# Modelos de Densidade Explícita



- Estratégias para garantir a tratabilidade dos modelos:
  - Construção cuidadosa de modelos que tenham estruturas pensadas para garantir sua tratabilidade
  - Modelos que admitam aproximações tratáveis da MLE e de seus gradientes

# Taxonomia de Modelos Generativos



**Figura:** Taxonomia dos modelos generativos [1].

# Modelos Tratáveis de Densidade Explícita



- PixelRNN [9]

- $p(x) = \prod_{i=1}^{n^2} p(x_i | x_1, \dots, x_{i-1})$

# Modelos Tratáveis de Densidade Explícita



- PixelRNN [9]

- $p(x) = \prod_{i=1}^{n^2} p(\textcolor{red}{x}_i | x_1, \dots, x_{i-1})$

- Pixel atual

# Modelos Tratáveis de Densidade Explícita



- PixelRNN [9]

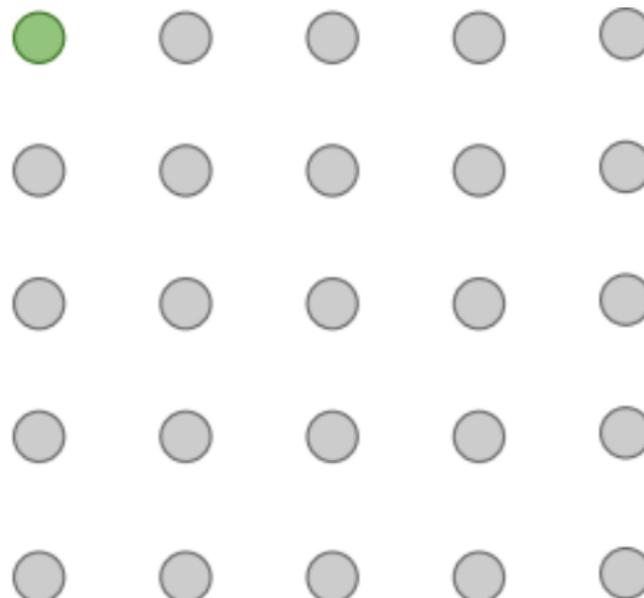
- $p(x) = \prod_{i=1}^{n^2} p(x_i | x_1, \dots, x_{i-1})$

- Pixels anteriores (condicional do modelo)

# Modelos Tratáveis de Densidade Explícita



- PixelRNN [9]

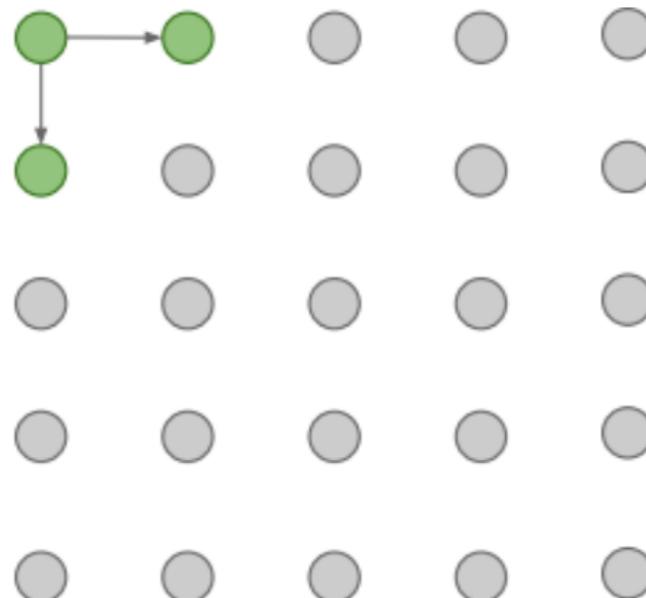


**Figura:** Geração Sequencial de Pixels numa PixelRNN.

# Modelos Tratáveis de Densidade Explícita



- PixelRNN [9]

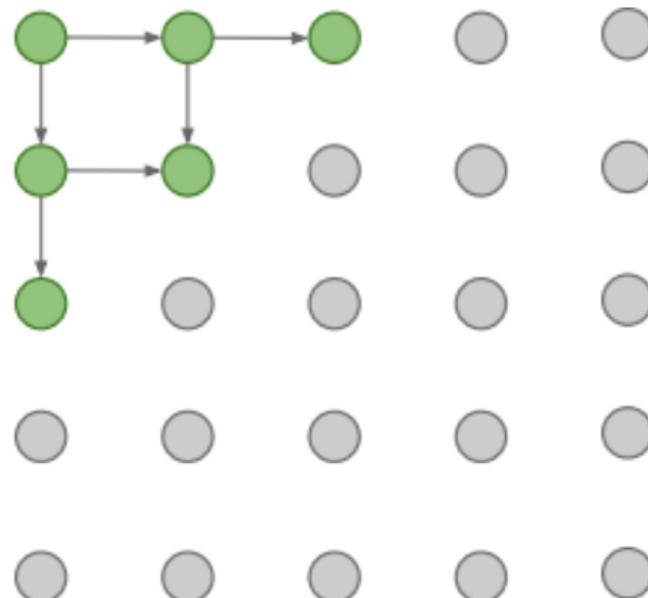


**Figura:** Geração Sequencial de Pixels numa PixelRNN.

# Modelos Tratáveis de Densidade Explícita



- PixelRNN [9]

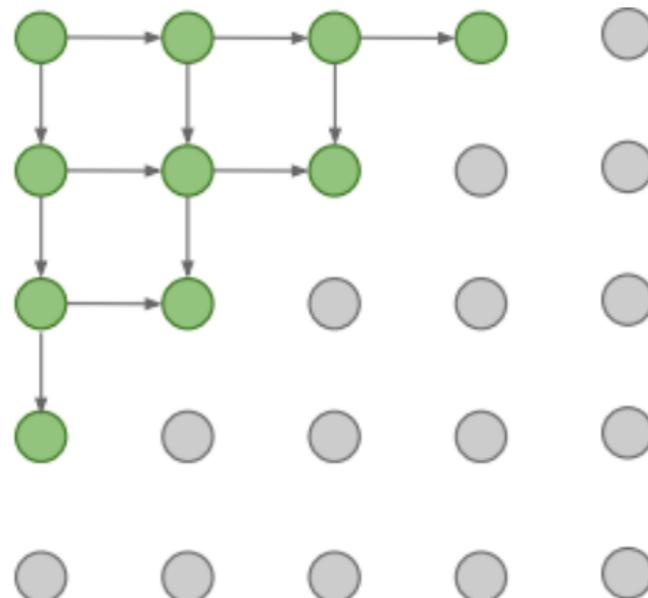


**Figura:** Geração Sequencial de Pixels numa PixelRNN.



# Modelos Tratáveis de Densidade Explícita

- PixelRNN [9]

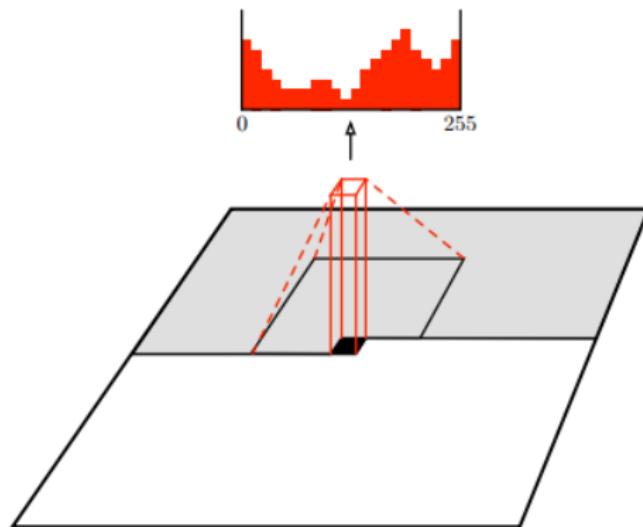


**Figura:** Geração Sequencial de Pixels numa PixelRNN.



# Modelos Tratáveis de Densidade Explícita

- PixelCNN [10]

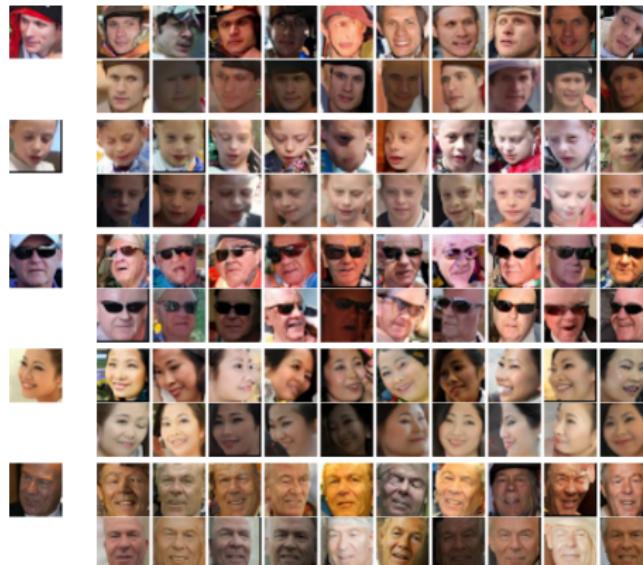


**Figura:** Geração Sequencial de Pixels numa PixelCNN.

# Modelos Tratáveis de Densidade Explícita



- Resultados de PixelRNNs e PixelCNNs



**Figura:** Samples geradas por meio de PixelCNNs.

# Modelos Tratáveis de Densidade Explícita



- PixelCNNs e PixelRNNs

- Prós:

- É possível computar a verossimilhança do modelo analiticamente
- Boas métricas matemáticas de avaliação (verossimilhanças altas)

- Contras:

- A geração pixel-a-pixel de amostras é um processo muito lento
- Geração de amostras não-realistas devido à falta de contexto global [6]

# Modelos Tratáveis de Densidade Explícita



- PixelCNNs e PixelRNNs
- Prós:
  - É possível computar a verossimilhança do modelo analiticamente
  - Boas métricas matemáticas de avaliação (**verossimilhanças altas**)
- Contras:
  - A geração pixel-a-pixel de amostras é um **processo muito lento**
  - Geração de **amostras não-realistas** devido à falta de contexto global [6]



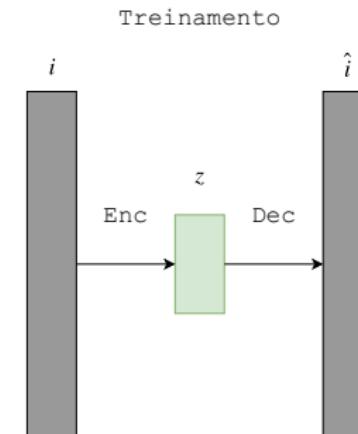
# Agenda

- 1** Introdução
- 2** Modelos Generativos
  - Taxonomia de Modelos Generativos
- 3** PixelRNN and PixelCNN
- 4** AutoEncoders
  - Variational AutoEncoder (VAE)
- 5** Generative Adversarial Networks
  - Histórico das GANs

# AutoEncoders



- Input:  $i$
- Vetor Latente:  $z$
- Output:  $\hat{i}$
- Loss:  $\| i - \hat{i} \| ^2$

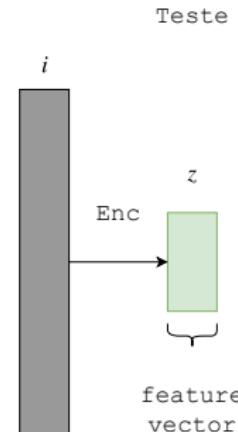


**Figura:** Procedimento de **treinamento** em um AutoEncoder.

# AutoEncoders



- Input:  $i$
- Vetor Latente:  $z$
- Output:  $\hat{i}$
- Loss:  $\| i - \hat{i} \| ^2$

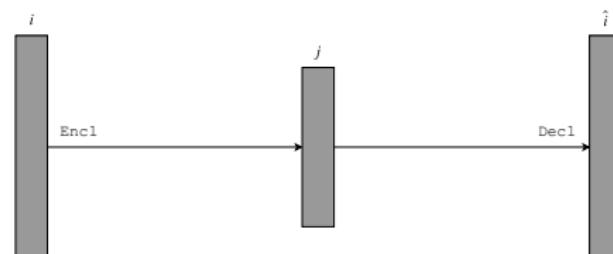


**Figura:** Procedimento de teste em um AutoEncoder.

# AutoEncoders



- Input:  $i$
- Vetor Latente:  $z$
- Output:  $\hat{i}$
- Loss:  $\| i - \hat{i} \| ^2$

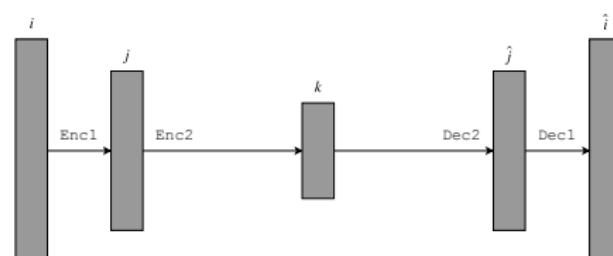


**Figura:** Arquitetura de um AutoEncoder com 1 camada.



# AutoEncoders

- Input:  $i$
- Inputs parciais:  $j$
- Vetor Latente:  $z$
- Output:  $\hat{i}$
- Outputs parciais:  $\hat{j}$
- Loss:  $\| i - \hat{i} \| ^2$

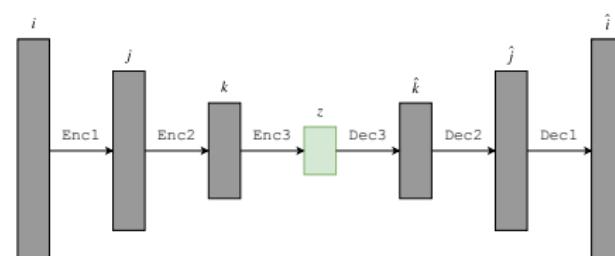


**Figura:** Arquitetura de um Stacked AutoEncoder com 2 camadas.

# AutoEncoders



- Input:  $i$
- Inputs parciais:  $j, k$
- Vetor Latente:  $z$
- Output:  $\hat{i}$
- Outputs parciais:  $\hat{j}, \hat{k}$
- Loss:  $\| i - \hat{i} \| ^2$



**Figura:** Arquitetura de um Stacked AutoEncoder com 3 camadas.



# AutoEncoders

- Arquiteturas Encoder-Decoder
- Treinado usando **stacking** ao invés de backpropagation
- Treino não-supervisionado
- Tarefas de reconstrução
- Loss Functions usadas em AEs compararam a saída do dado com o próprio dado
  - MSE
  - L1
  - KLDiv

# AutoEncoders

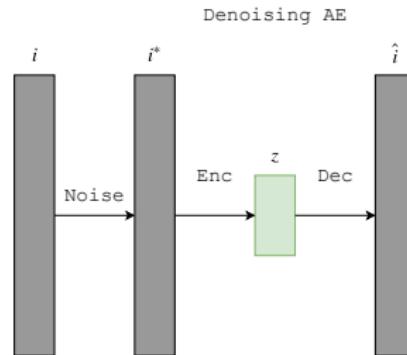


- Usados para várias tasks
  - Compressão de Dados
  - Redução de Dimensionalidade
  - Denoising
  - Sparse Coding
  - Geração de Amostras (Variational AutoEncoder) [11]
  - Arquiteturas adversariais [4, 12]

# AutoEncoders

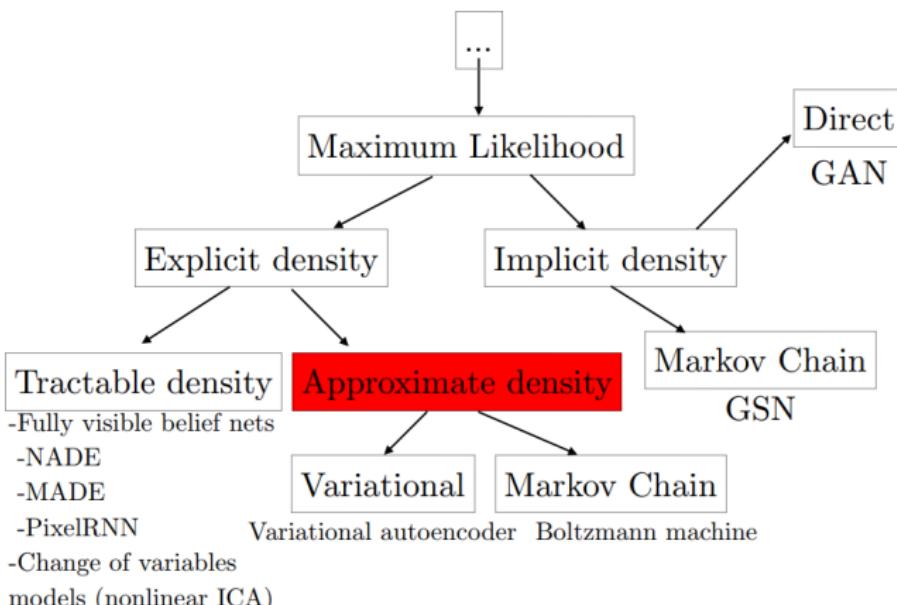


- Input:  $i$
- Input com ruído:  $i^*$
- Vetor Latente:  $z$
- Output:  $\hat{i}$
- Loss:  $\| i - \hat{i} \|_2^2$



**Figura:** Arquitetura de um Denoising AutoEncoder.

# Taxonomia de Modelos Generativos



**Figura:** Taxonomia dos modelos generativos [1].

# Modelos de Densidade Explícita



- Estratégias para garantir a tratabilidade dos modelos:
  - Construção cuidadosa de modelos que tenham estruturas pensadas para garantir sua tratabilidade
  - Modelos que admitam aproximações tratáveis da MLE e de seus gradientes

# Variational AutoEncoder (VAE) [11]



- Geração de amostras
- Inferência variacional<sup>3</sup>
  - Limite Inferior da MLE
- Amostragem feita de uma **distribuição gaussiana**
  - Simples de implementar
  - VAE aprende a **mapear** a gaussiana para variáveis latentes do dado

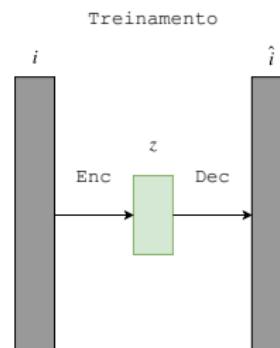
<sup>3</sup><http://pedrounb.blogspot.com/2017/12/introducao-inferencia-variacional.html>



# Variational AutoEncoder (VAE) [11]

## AE para Geração de Amostras

É possível treinar um AE para realizar compressão, remoção de ruído, sparse coding...



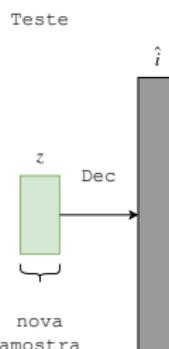
**Figura:** Geração de amostras em um AE.



# Variational AutoEncoder (VAE) [11]

## AE para Geração de Amostras

Será que é possível amostrar valores para o vetor  $z$  e gerar novas amostras?



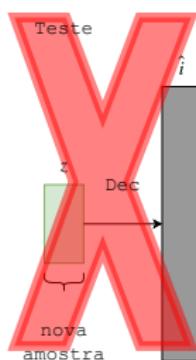
**Figura:** Geração de amostras em um AE.



# Variational AutoEncoder (VAE) [11]

## AE para Geração de Amostras

Será que é possível amostrar valores para o vetor  $z$  e gerar novas amostras?



**Figura:** Geração de amostras em um AE.

# Variational AutoEncoder (VAE) [11]



## AE para Geração de Amostras

AEs comuns oferecem **pouco controle sobre como as informações estão sendo representadas** em  $z$ .

# Variational AutoEncoder (VAE) [11]



## AE para Geração de Amostras

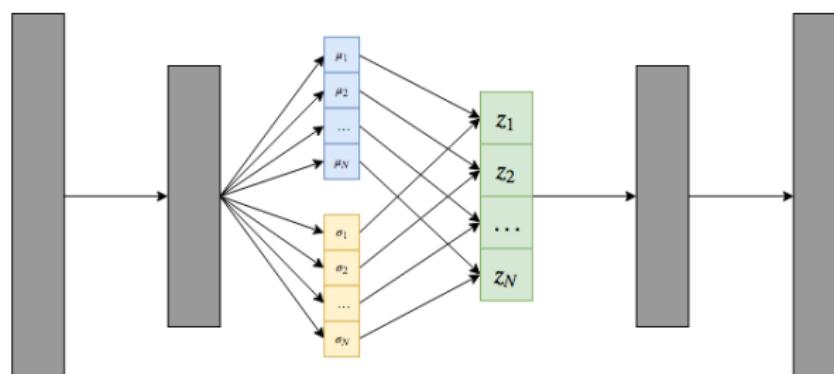
Variational AEs (VAEs) permitem um maior controle sobre  $z$  e, consequentemente, a geração de novas amostras por meio da amostragem de vetores latentes  $z$ .



# Variational AutoEncoder (VAE) [11]

## AE para Geração de Amostras

Para fazer uma amostragem de um valor  $z_i \in z$  de acordo com uma Distribuição Gaussiana ( $z_i \sim N(\mu_i, \sigma_i)$ ), é preciso otimizar os parâmetros  $\mu_i$  e  $\sigma_i$  dessa gaussiana.



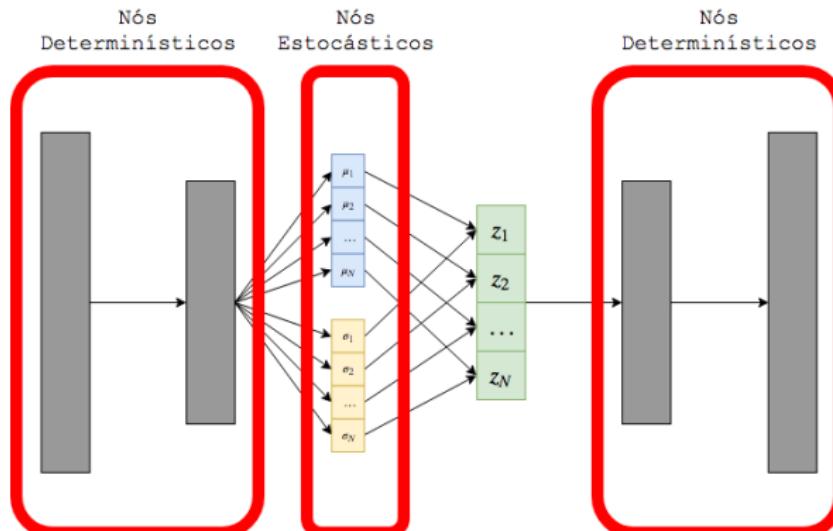
**Figura:** Geração de amostras em um AE.



# Variational AutoEncoder (VAE) [11]

## AE para Geração de Amostras

Como fazer o backpropagation através de valores amostrados de uma distribuição estatística?



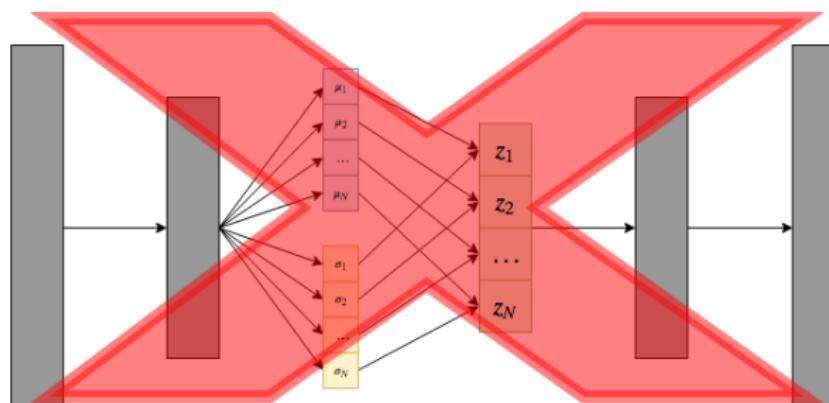
**Figura:** Geracão de amostras em um AE.



# Variational AutoEncoder (VAE) [11]

## AE para Geração de Amostras

Nessa arquitetura o backpropagation é impossível.



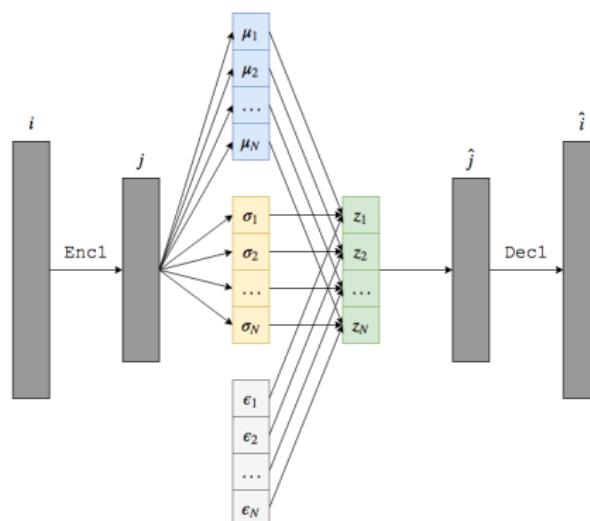
**Figura:** Geração de amostras em um AE.



# Variational AutoEncoder (VAE) [11]

## AE para Geração de Amostras

Para permitir o backpropagation é usado o truque da **reparametrização**.



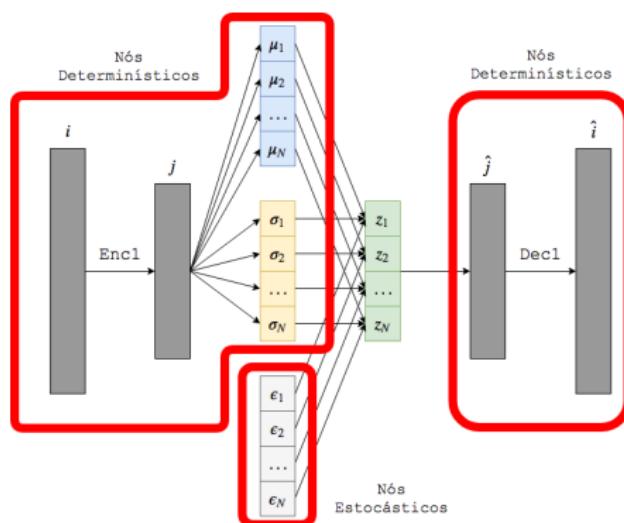
**Figura:** Geração de amostras em um AE.



# Variational AutoEncoder (VAE) [11]

## AE para Geração de Amostras

Para permitir o backpropagation é usado o truque da **reparametrização**.



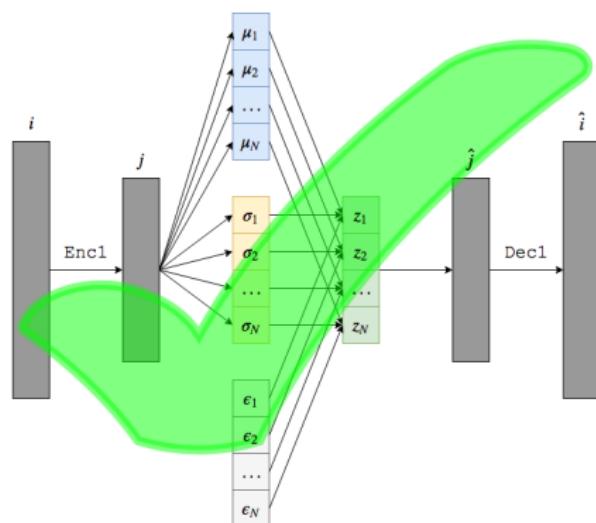
**Figura:** Geração de amostras em um AE.



# Variational AutoEncoder (VAE) [11]

## AE para Geração de Amostras

Para permitir o backpropagation é usado o truque da **reparametrização**.



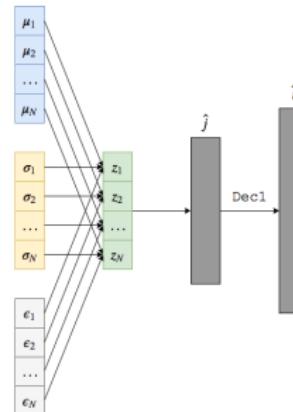
**Figura:** Geração de amostras em um AE.



# Variational AutoEncoder (VAE) [11]

## AE para Geração de Amostras

Depois de treinada a arquitetura, basta amostrar valores de  $\varepsilon_i, i = 1, 2, \dots, N$  para gerar amostras novas da distribuição que foi modelada.



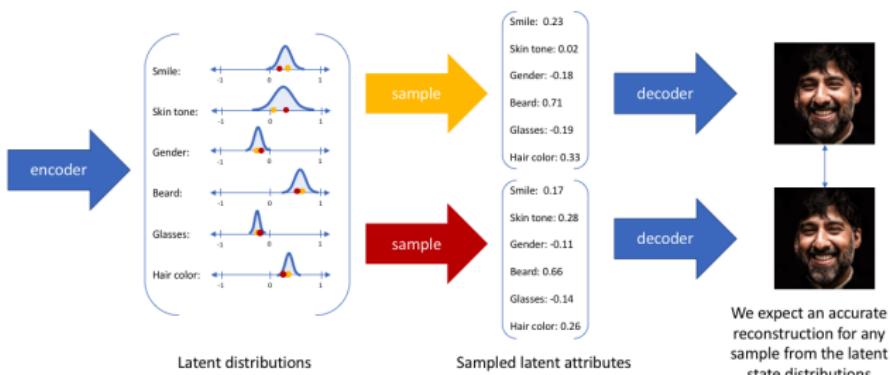
**Figura:** Geração de amostras em um AE.



# Variational AutoEncoder (VAE) [11]

## Interpretação do Vetor $z$

Os valores do vetor latente  $z$  tendem a mapear **informações de alto nível semântico** nos dados.



**Figura:** Gaussianas em um VAE<sup>4</sup>.

<sup>4</sup><https://www.jeremyjordan.me/variational-autoencoders/>

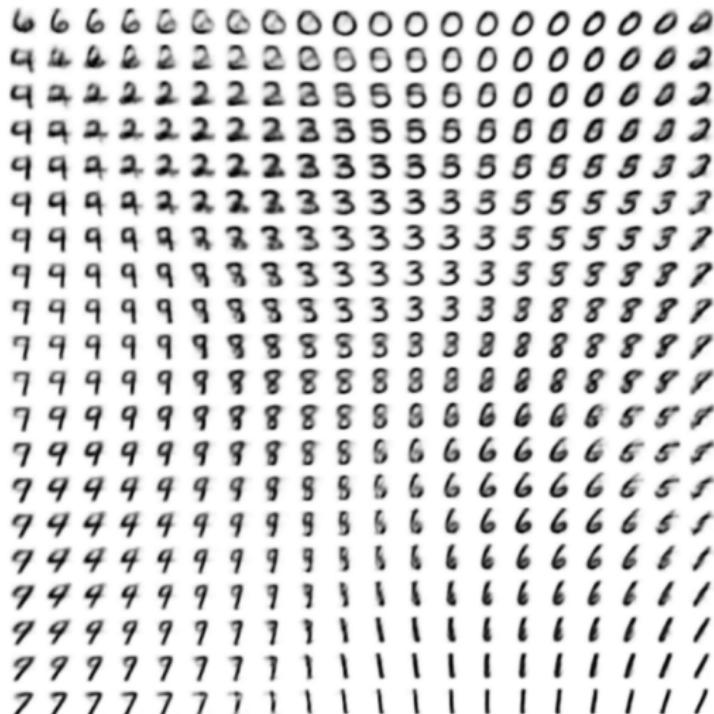
## Variational AutoEncoder (VAE) [11]



**Figura:** Variando o espaço de  $z$  em um VAE.



## Variational AutoEncoder (VAE) [11]



**Figura:** Variando o espaço de  $z$  em um VAE.

## Convolutional AE



Exercício Prático/Demo - Convolutional AE

**AE\_Convolutional.ipynb**

# Denoising AE



Exercício Prático/Demo - Denoising AE

**AE\_Denoising.ipynb**

# Sparse AE



Exercício Prático/Demo - Sparse AE

**AE\_Sparse.ipynb**

## Variational AEs



Demo - VAEs

**AE\_Variational.ipynb**

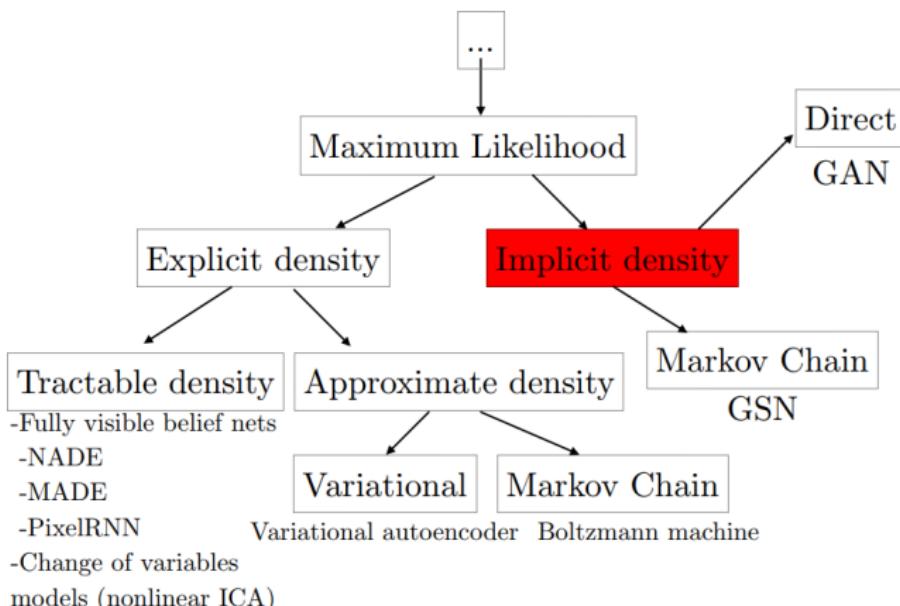


# Agenda

- 1** Introdução
- 2** Modelos Generativos
  - Taxonomia de Modelos Generativos
- 3** PixelRNN and PixelCNN
- 4** AutoEncoders
  - Variational AutoEncoder (VAE)
- 5** Generative Adversarial Networks
  - Histórico das GANs



# Taxonomia de Modelos Generativos



**Figura:** Taxonomia dos modelos generativos [1].

# Generative Adversarial Networks



- Extração de deep features não supervisionada
- Aprendizado indireto da distribuição dos dados
- Duas redes com objetivos opostos
- Convergência conjunta
- Background matemático de teoria dos jogos



# Generative Adversarial Networks

- Rede Discriminativa ( $D$ )

- Treinada para discernir entre amostras reais do dataset de treino e amostras falsas vindas de  $G$
- Tem como objetivo discriminar entre amostras reais e falsas

- Rede Generativa ( $G$ )

- Mapeia a distribuição de um vetor latente  $Z$  para a distribuição das amostras de treino
- Tem o objetivo de criar amostras cada vez mais verossímeis



# Generative Adversarial Networks

- As funções de perda de cada uma das redes devem ser cuidadosamente desenhadas
  - Minimax Loss (MM GAN)
  - Non-Saturating Loss (NS GAN)
  - Wasserstein Distance (WGAN)
  - Mínimos Quadrados (Least Squares – LSGAN)
  - ...
- GAN Zoo
  - <https://github.com/hindupuravinash/the-gan-zoo>

# Minimax



- Função de Custo da Rede Discriminativa

- $C^{(D)} = -\frac{1}{2}\mathbb{E}_{x \sim p_{data}} \log D(x) - \frac{1}{2}\mathbb{E}_z \log (1 - D(G(z)))$

- Função de Custo da Rede Generativa

- $C^{(G)} = -C^{(D)}$



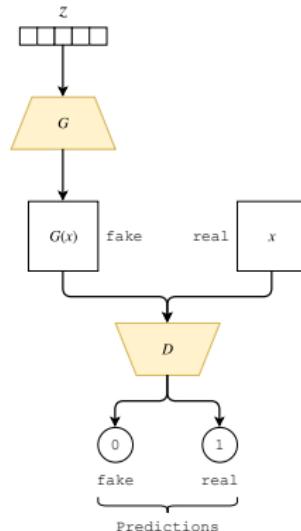
# Treinamento de uma GAN

## Treinamento de uma GAN

As funções de perda  $\mathcal{L}_G$  e  $\mathcal{L}_D$  de  $G$  e  $D$  possuem objetivos contrários. Por isso, o “jogo” entre as duas redes é classificado como um **cenário adversarial**.



# Treinamento de uma GAN



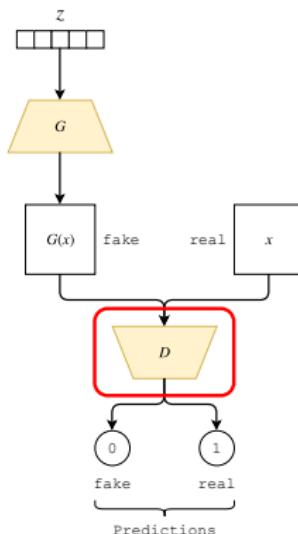
**Figura:** Arquitetura de uma GAN tradicional.



# Treinamento de uma GAN

## Treinamento de uma GAN

$D$  quer acertar a classificação entre amostras reais e sintéticas.



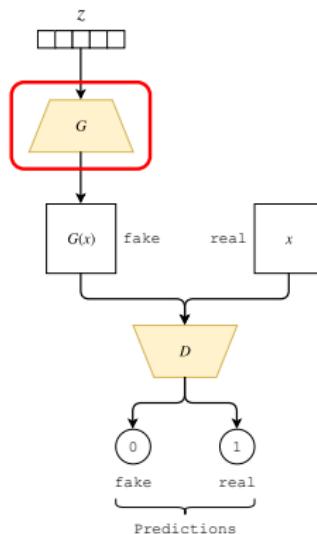
**Figura:** Arquitetura de uma GAN tradicional.



# Treinamento de uma GAN

## Treinamento de uma GAN

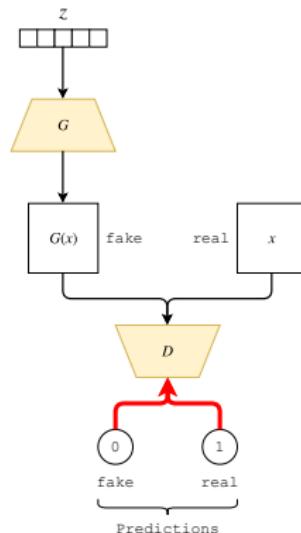
$G$  quer fazer  $D$  errar a classificação entre amostras reais e sintéticas.



**Figura:** Arquitetura de uma GAN tradicional.



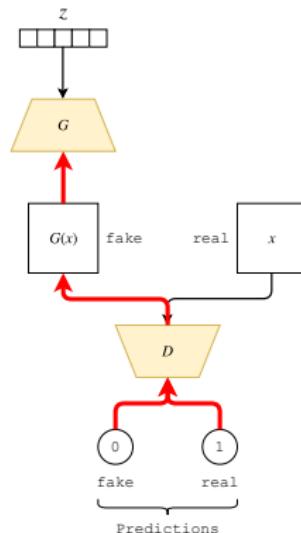
# Treinamento de uma GAN



**Figura:** Backpropagation em uma GAN tradicional.

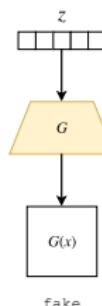


# Treinamento de uma GAN



**Figura:** Backpropagation em uma GAN tradicional.

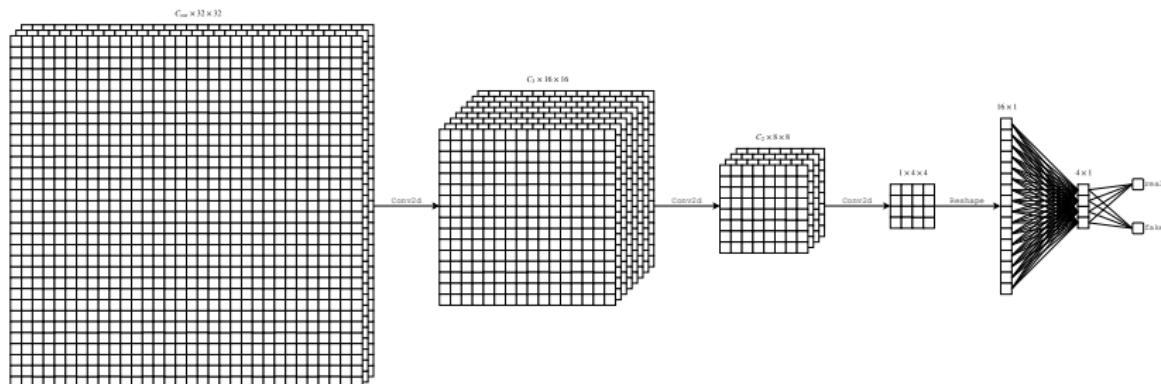
# Treinamento de uma GAN



**Figura:** Geração de imagens em uma GAN tradicional.



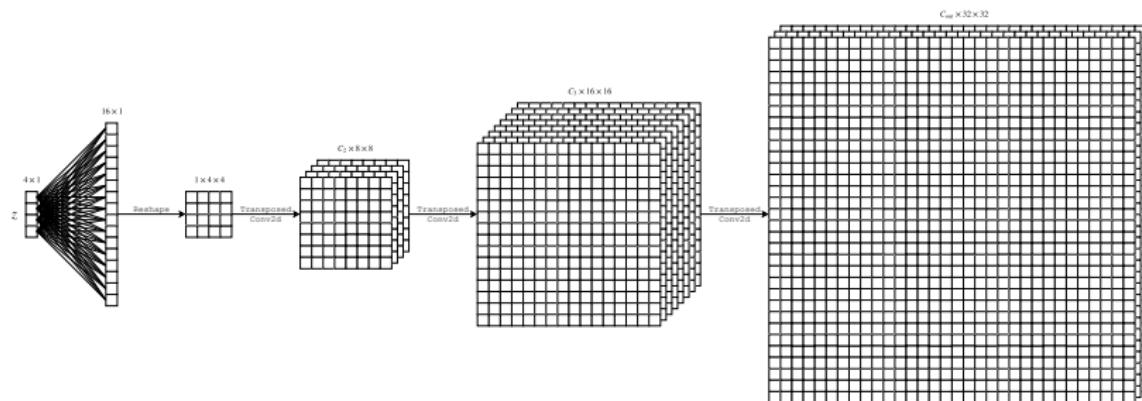
# Treinamento de uma GAN



**Figura:** Arquitetura de  $D$ .



# Treinamento de uma GAN



**Figura:** Arquitetura de  $G$ .



# Dificuldades de Treinamento

## Dificuldades de Treinamento

GANs **não possuem uma única função de loss** como é o caso de arquiteturas tradicionais como MLPs, CNNs, FCNs, RNNs etc. Ao invés disso, o que é otimizado é o **objetivo composto** do conjunto das losses de  $G$  e de  $D$ .



# Dificuldades de Treinamento

## Dificuldades de Treinamento

Como os objetivos das losses é contrário, é possível treinar  $G$  e  $D$  conjuntamente de forma não supervisionada. Convergência é atingida quando se chega no **Equilíbrio de Nash<sup>5</sup> <sup>6</sup>** [13].

---

<sup>5</sup> [https://medium.com/@jonathan\\_hui/gan-why-it-is-so-hard-to-train-generative-advisory-networks-819a86b3750b](https://medium.com/@jonathan_hui/gan-why-it-is-so-hard-to-train-generative-advisory-networks-819a86b3750b)

<sup>6</sup> <https://ahmednabibrahim.wordpress.com/2017/01/17/generative-adversarial-networks-when-deep-learning-meets-game-theory/>

# Dificuldades de Treinamento



## Dificuldades de Treinamento

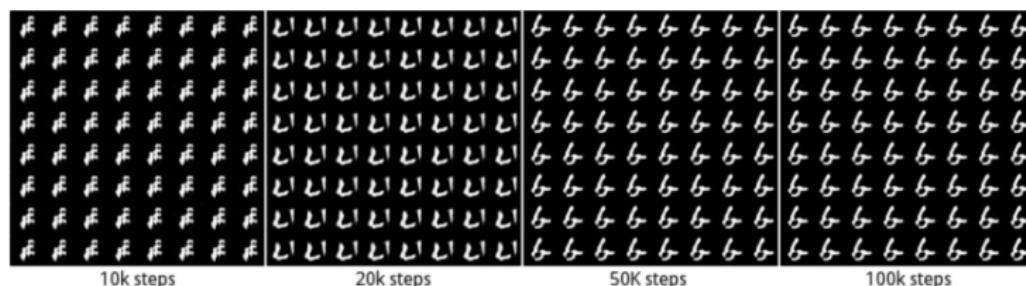
Erros de balanceamento nas capacidades de parâmetros de  $G$  e  $D$  podem causar uma rede a ficar muito melhor que a outra no “jogo”. Quando isso acontece, a **convergência do conjunto  $\{D, G\}$  fica comprometida**.



# Dificuldades de Treinamento

## Dificuldades de Treinamento

GANs são suscetíveis a alguns problemas de treinamento originários da sua natureza adversarial. O mais comum entre eles é o **Mode Collapse**, no qual  $G$  aprende a gerar amostras com pouquíssima variação, mas que enganam bem  $D$ . Isso efetivamente **impede  $G$  de mapear corretamente** todo o espectro de valores da distribuição que gera os dados de treino.



**Figura:** Mode Collapse numa GAN treinada no MNIST. Fonte: [14].



# Dificuldades de Treinamento

## Dificuldades de Treinamento

Por esses motivos, GANs são consideradas redes **bem mais difíceis de se treinar que redes tradicionais.**



# Dificuldades de Treinamento

## MiniMax Loss (MM GAN)

$$C^{(D)} = -\frac{1}{2} \mathbb{E}_{x \sim p_{data}} \log D(x) - \frac{1}{2} \mathbb{E}_z \log (1 - D(G(z)))$$

$$C^{(G)} = -C^{(D)}$$

## MM GAN

É possível representar as duas losses com uma única função.



# Dificuldades de Treinamento

## Dificuldades de Treinamento

Já que todo gradiente que chega em  $G$  no backpropagation passa primeiramente por  $D$ , o discriminador tende a ficar bom na sua tarefa mais rapidamente. Além disso, no começo do treinamento  $G$  gera essencialmente imagens aleatórias, facilitando a tarefa de  $D$ . Esse problema se chama de Saturação do Gradiente, já que, como  $D$  acerta a classe (real ou fake) de quase todas as amostras da classificação, não há mais erros para propagar o gradiente para  $G$ .



# Dificuldades de Treinamento

## Non-Saturating Loss (NS GAN)

$$C^{(D)} = -\frac{1}{2} \mathbb{E}_{x \sim p_{data}} \log D(x) - \frac{1}{2} \mathbb{E}_z \log (1 - D(G(z)))$$

$$C^{(G)} = -\frac{1}{2} \mathbb{E}_z \log D(G(z))$$

## NS GAN

NS GANs não sofrem do problema de Saturação do Gradiente de  $D$ .



# Dificuldades de Treinamento

## Dificuldades de Treinamento

Avanços em GANs hoje em dia são focados no design de novas arquiteturas e, principalmente, de **loss functions com características distintas** que sejam desejáveis para o treinamento.

GAN	DISCRIMINATOR LOSS	GENERATOR LOSS
MM GAN	$\mathcal{L}_D^{GAN} = -\mathbb{E}_{x \sim p_d}[\log(D(x))] - \mathbb{E}_{\hat{x} \sim p_g}[\log(1 - D(\hat{x}))]$	$\mathcal{L}_G^{GAN} = \mathbb{E}_{\hat{x} \sim p_g}[\log(1 - D(\hat{x}))]$
NS GAN	$\mathcal{L}_D^{NSGAN} = -\mathbb{E}_{x \sim p_d}[\log(D(x))] - \mathbb{E}_{\hat{x} \sim p_g}[\log(1 - D(\hat{x}))]$	$\mathcal{L}_G^{NSGAN} = -\mathbb{E}_{\hat{x} \sim p_g}[\log(D(\hat{x}))]$
WGAN	$\mathcal{L}_D^{WGAN} = -\mathbb{E}_{x \sim p_d}[D(x)] + \mathbb{E}_{\hat{x} \sim p_g}[D(\hat{x})]$	$\mathcal{L}_G^{WGAN} = -\mathbb{E}_{\hat{x} \sim p_g}[D(\hat{x})]$
WGANGP	$\mathcal{L}_D^{WGANGP} = \mathcal{L}_D^{WGAN} + \lambda \mathbb{E}_{\hat{x} \sim p_g}[(\ \nabla D(\alpha x + (1 - \alpha)\hat{x})\ _2 - 1)^2]$	$\mathcal{L}_G^{WGANGP} = -\mathbb{E}_{\hat{x} \sim p_g}[D(\hat{x})]$
LS GAN	$\mathcal{L}_D^{LSGAN} = -\mathbb{E}_{x \sim p_d}[(D(x) - 1)^2] + \mathbb{E}_{\hat{x} \sim p_g}[D(\hat{x})^2]$	$\mathcal{L}_G^{LSGAN} = -\mathbb{E}_{\hat{x} \sim p_g}[(D(\hat{x} - 1)^2)]$
DRAGAN	$\mathcal{L}_D^{DRAGAN} = \mathcal{L}_D^{GAN} + \lambda \mathbb{E}_{\hat{x} \sim p_d + \mathcal{N}(0, c)}[(\ \nabla D(\hat{x})\ _2 - 1)^2]$	$\mathcal{L}_G^{DRAGAN} = \mathbb{E}_{\hat{x} \sim p_g}[\log(1 - D(\hat{x}))]$
BEGAN	$\mathcal{L}_D^{BEGAN} = \mathbb{E}_{x \sim p_d}[\ x - AE(x)\ _1] - k_t \mathbb{E}_{\hat{x} \sim p_g}[\ \hat{x} - AE(\hat{x})\ _1]$	$\mathcal{L}_G^{BEGAN} = \mathbb{E}_{\hat{x} \sim p_g}[\ \hat{x} - AE(\hat{x})\ _1]$

**Figura:** Loss functions de diferentes tipos de GANs. Fonte: [15].

# Dificuldades de Treinamento



## Dificuldades de Treinamento

Entre 2014 e 2016, vários trabalhos focaram na **melhoria da estabilidade** de treinamento e em boas práticas que ajudam na convergência de GANs [16, 17].

## Histórico das GANs



- GANs [13] tradicionais
- Conditional GANs (CGANs) [18]
- Deep Convolutional GANs (DCGANs) [16]
- infoGANs [19]
- Progressive GANs [20]
- ...



## Histórico das GANs

- GANs [13] tradicionais
  - Fully Connected
  - Limitadas a imagens pequenas
  - Qualidade visual limitada

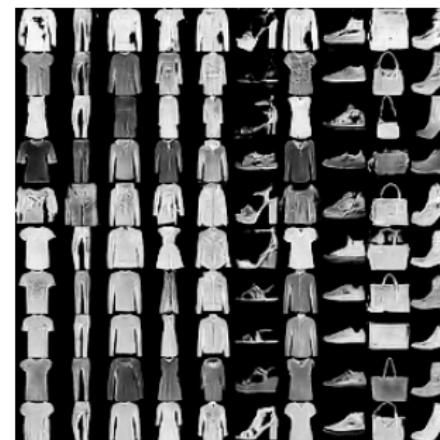


# Histórico das GANs



- CGANs [18]

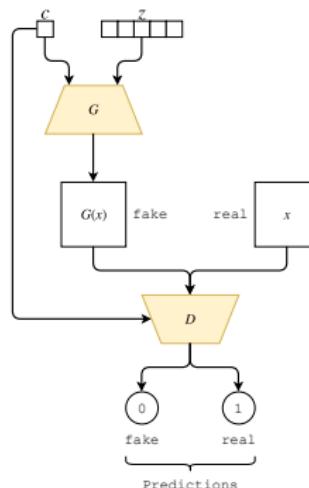
- Fully Connected
- Condiciona a geração de amostras por classes no dataset
- One hot encoding





# Histórico das GANs

- CGANs [18]



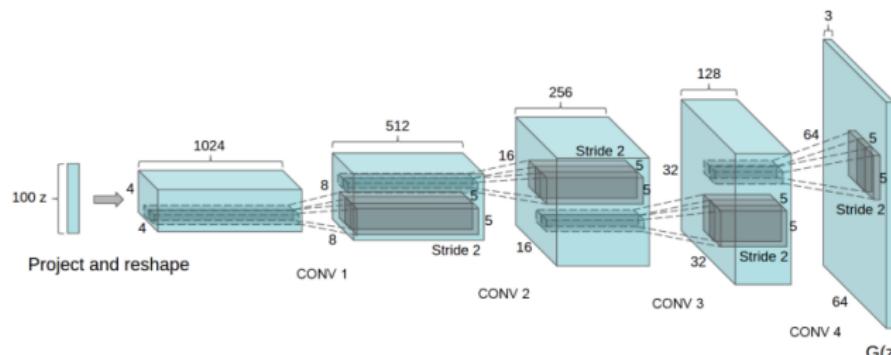
**Figura:** Arquitetura de uma CGAN.



# Histórico das GANs

- DCGANs [16]

- Arquitetura Convolucional
- Tamanho das imagens não é mais uma limitação tão grande
- Interpolações em  $Z$  com interpretabilidade semântica
- Melhor qualidade visual



**Figura:** Arquitetura de uma DCGAN. Fonte: [16].

## Histórico das GANs



- DCGANs [16]

- Arquitetura Convolucional
- Tamanho das imagens não é mais uma limitação tão grande
- Interpolações em  $Z$  com interpretabilidade semântica
- Melhor qualidade visual



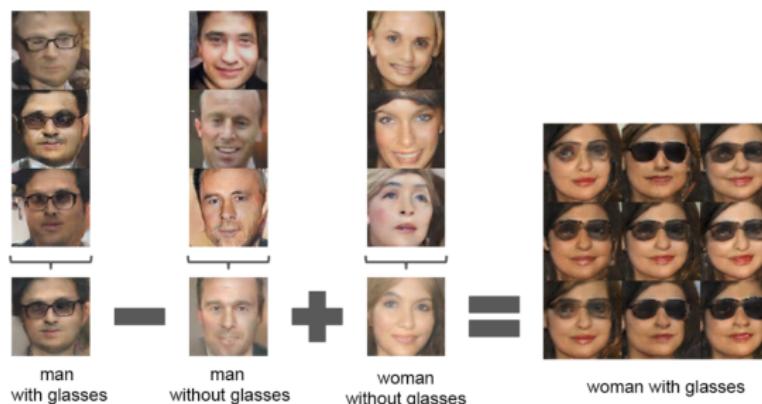
**Figura:** Amostras geradas por DCGANs. Fonte: [16].



## Histórico das GANs

- DCGANs [16]

- Arquitetura Convolucional
- Tamanho das imagens não é mais uma limitação tão grande
- Interpolações em  $Z$  com interpretabilidade semântica
- Melhor qualidade visual



**Figura:** Aritmética de features em  $Z$  e seu efeito nas amostras geradas. Fonte: [16].

# Histórico das GANs



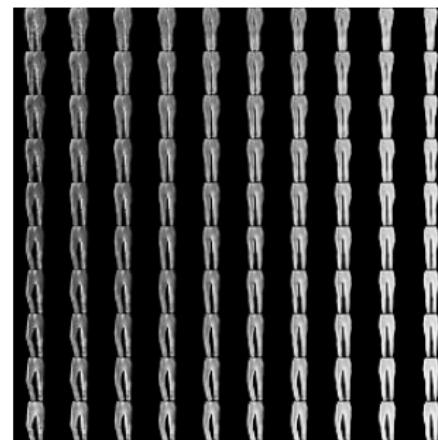
- De GANs [13] para DCGANs [16]
  - Fazer downsampling usando strides maiores nas convoluções ao invés de max-pooling
  - Usar Convolução Transposta (Deconvolução) para os upsamplings
  - Eliminar as camadas Fully Connected
  - Usar Batch Normalization em todas as camadas, exceto na camada de output da generativa e na camada de input da discriminativa
  - Usar ReLU na generativa, exceto no output que usa uma ativação Tanh
  - Usar ativações LeakyReLU na rede discriminativa

# Histórico das GANs



- infoGANs [19]
  - Maior controle sobre o espaço de features de entrada (vetor de ruído)

8	8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8	8

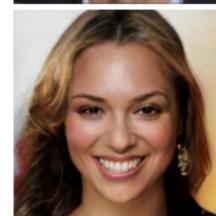
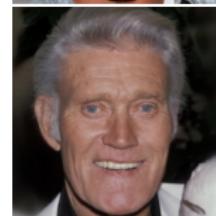
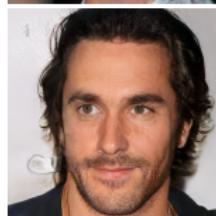




## Histórico das GANs

- Progressive GANs [20]

- Qualidade visual impressionante para imagens com maiores resoluções
- Parte de resoluções baixas e vai iterativamente adicionando camadas no gerador e discriminador
- Estabilidade do treinamento melhorada para imagens maiores



# Convolutional GANs



Exercício Prático/Demo - Convolutional GANs

**GAN\_Convolutional.ipynb**

# Conditional GANs



Exercício Prático/Demo - Conditional GANs

**GAN\_Conditional.ipynb**

# GANs Diversas



Exercício Prático/Demo - GANs Diversas

**Pytorch\_GANs.ipynb**

## References

- [1] Ian Goodfellow.  
Nips 2016 tutorial: Generative adversarial networks.  
[arXiv preprint arXiv:1701.00160](https://arxiv.org/abs/1701.00160), 2016.
- [2] Chelsea Finn, Ian Goodfellow, and Sergey Levine.  
Unsupervised learning for physical interaction through video prediction.  
In *Advances in neural information processing systems*, pages 64–72, 2016.
- [3] Chelsea Finn and Sergey Levine.  
Deep visual foresight for planning robot motion.  
In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 2786–2793. IEEE, 2017.
- [4] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz.  
Multimodal unsupervised image-to-image translation.  
[arXiv preprint arXiv:1804.04732](https://arxiv.org/abs/1804.04732), 2018.
- [5] William Lotter, Gabriel Kreiman, and David Cox.  
Unsupervised learning of visual structure using predictive generative networks.  
[arXiv preprint arXiv:1511.06380](https://arxiv.org/abs/1511.06380), 2015.
- [6] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros.  
Image-to-image translation with conditional adversarial networks.  
[arXiv preprint](https://arxiv.org/abs/1611.07004), 2017.
- [7] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros.  
Generative visual manipulation on the natural image manifold.  
In *European Conference on Computer Vision*, pages 597–613. Springer, 2016.
- [8] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston.  
Neural photo editing with introspective adversarial networks.  
[arXiv preprint arXiv:1609.07093](https://arxiv.org/abs/1609.07093), 2016.
- [9] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu.  
Pixel recurrent neural networks.  
[arXiv preprint arXiv:1601.06759](https://arxiv.org/abs/1601.06759), 2016.
- [10] Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al.



## References

- Conditional image generation with pixelcnn decoders.  
In [Advances in Neural Information Processing Systems](#), pages 4790–4798, 2016.
- [11] Diederik P Kingma and Max Welling.  
Auto-encoding variational bayes.  
[arXiv preprint arXiv:1312.6114](#), 2013.
- [12] Ming-Yu Liu, Thomas Breuel, and Jan Kautz.  
Unsupervised image-to-image translation networks.  
In [Advances in Neural Information Processing Systems](#), pages 700–708, 2017.
- [13] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio.  
Generative adversarial nets.  
In [Advances in neural information processing systems](#), pages 2672–2680, 2014.
- [14] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein.  
Unrolled generative adversarial networks.  
[arXiv preprint arXiv:1611.02163](#), 2016.
- [15] Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet.  
Are gans created equal? a large-scale study.  
[arXiv preprint arXiv:1711.10337](#), 2017.
- [16] Alec Radford, Luke Metz, and Soumith Chintala.  
Unsupervised representation learning with deep convolutional generative adversarial networks.  
[arXiv preprint arXiv:1511.06434](#), 2015.
- [17] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen.  
Improved techniques for training gans.  
In [Advances in Neural Information Processing Systems](#), pages 2234–2242, 2016.
- [18] Mehdi Mirza and Simon Osindero.  
Conditional generative adversarial nets.  
[arXiv preprint arXiv:1411.1784](#), 2014.
- [19] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel.  
Infogan: Interpretable representation learning by information maximizing generative adversarial nets.



## └ References

In Advances in neural information processing systems, pages 2172–2180, 2016.

- [20] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen.  
Progressive growing of gans for improved quality, stability, and variation.  
arXiv preprint arXiv:1710.10196, 2017.

