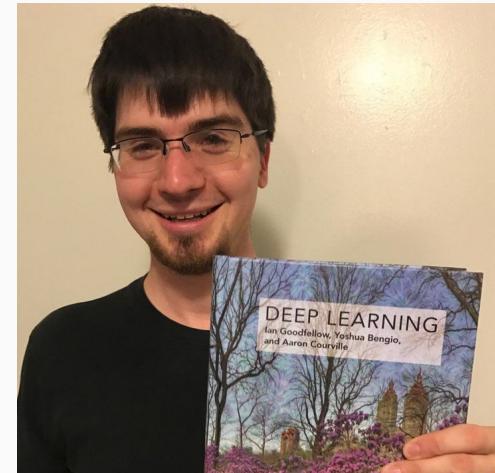


Deep Learning Generative Adversarial Networks - GANs

Renato Assunção - ESRI Inc. and DCC/UFMG

GAN - história bem curta

- GAN = Generative Adversarial Network
- Rede Generativa (ou Gerativa) Adversária
- Introduzidas em NIPS 2014.
- Ian Goodfellow + Yoshua Bengio + outros da Univ. de Montreal.
- Yann LeCun, diretor de pesquisa em AI do Facebook:
 - “a ideia mais interessante dos últimos 10 anos em ML”

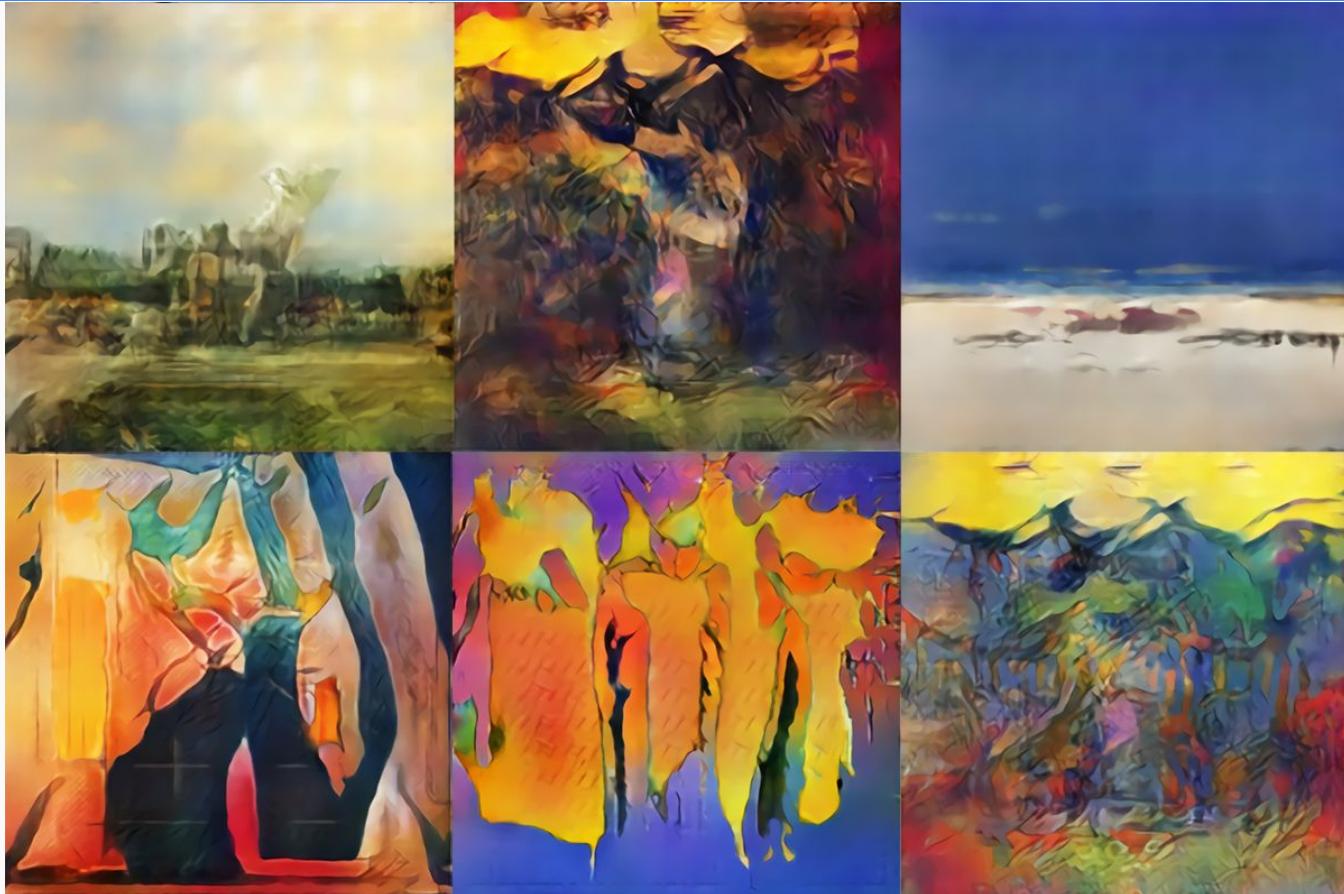


Generative Adversarial Nets

Ian J. Goodfellow, Jean Pouget-Abadie*, Mehdi Mirza, Bing Xu, David Warde-Farley,
Sherjil Ozair†, Aaron Courville, Yoshua Bengio†
Département d'informatique et de recherche opérationnelle
Université de Montréal
Montréal, QC H3C 3J7



Seu computador é um artista



*Art and
Artificial
Intelligence
Laboratory,
Rutgers
University*

O computador é um artista

- Christie's é uma casa especializada em leilões.
- Ela vendeu uma pintura gerada por uma GAN por US \$ 432.000
- GAN baseada em código aberto escrito por Robbie Barrat, de Stanford.
- Como a maioria dos artistas de verdade, ele não viu nada do dinheiro, que foi para a empresa francesa, Obvious.

Christie's sells its first AI portrait for \$432,500, beating estimates of \$10,000

The image was created using a machine learning algorithm that scanned historical artwork

By James Vincent | Oct 25, 2018, 1:03pm EDT

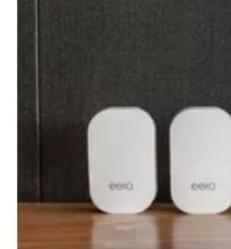
f t SHARE



GOOD



Here are the best Amazon deals so far



Fake Celeb Faces

- Criação de exemplos novos, não-vistos, inexistentes por simulação probabilística.
- Exemplos simulados são similares (mas diferentes) àqueles numa base de treinamento.



Figure 5: 1024×1024 images generated using the CELEBA-HQ dataset. See Appendix F for a larger set of results, and the accompanying video for latent space interpolations.

<https://arxiv.org/pdf/1710.10196.pdf>

Qual face é real?

<https://www.whichfaceisreal.com/>

Criar personagens de anime

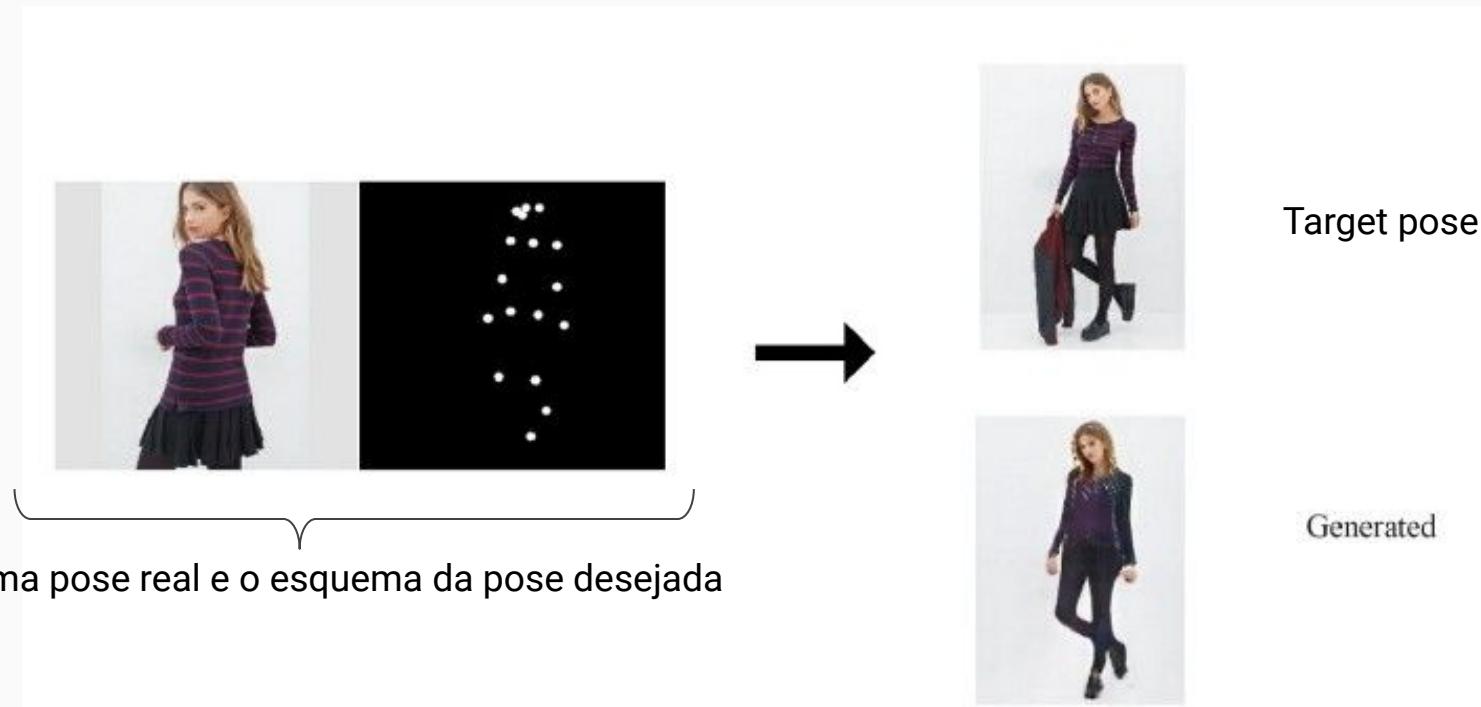
- O desenvolvimento de jogos e a produção de animação são caros.
- Contratam muitos artistas de produção para tarefas relativamente rotineiras.
- GAN pode gerar e colorir automaticamente os personagens do anime.



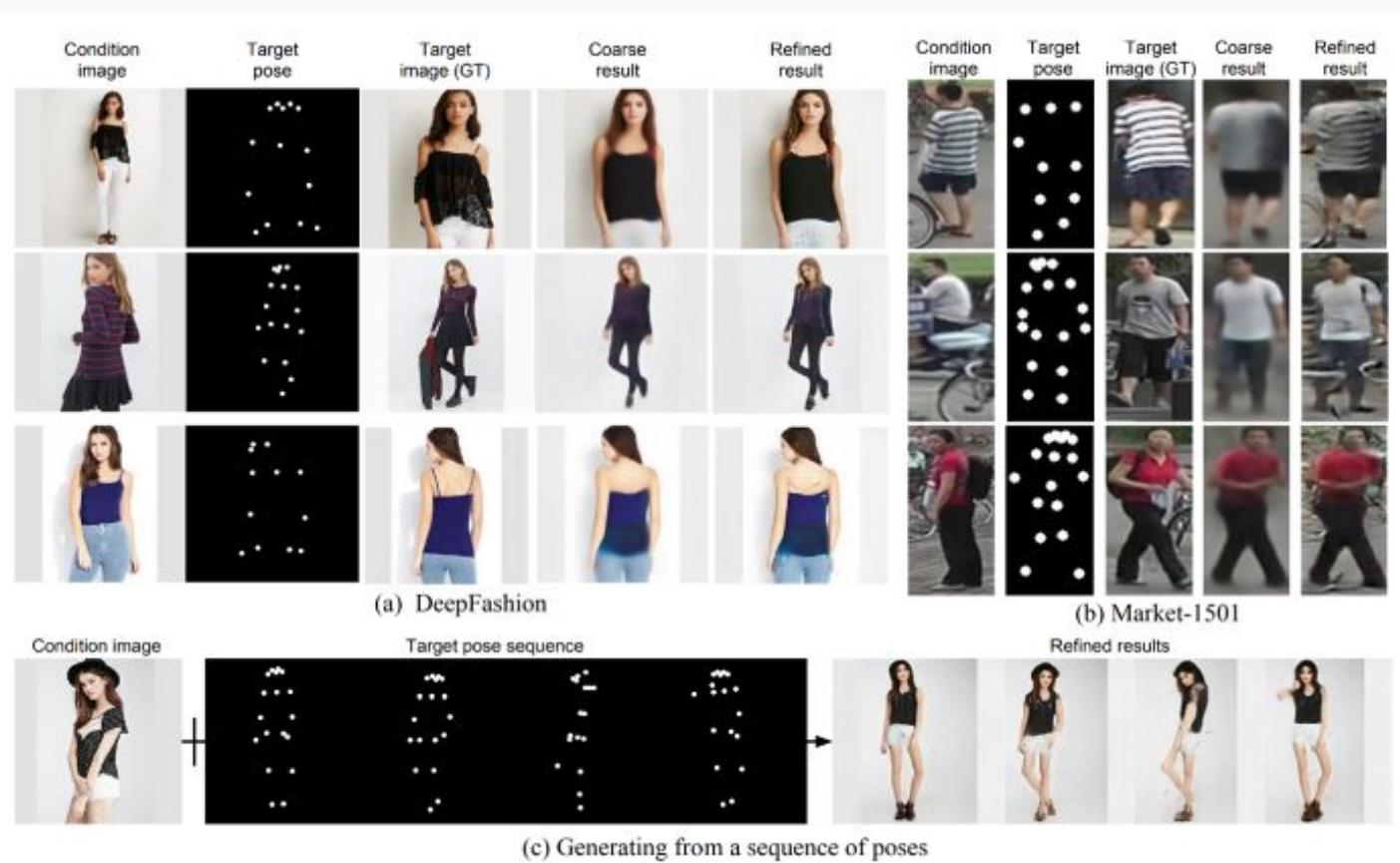
Figure 7: Generated samples

Geração de poses

- Input = uma pose
- Saída = mesma pessoa em outra pose



<https://papers.nips.cc/paper/6644-pose-guided-person-image-generation.pdf>

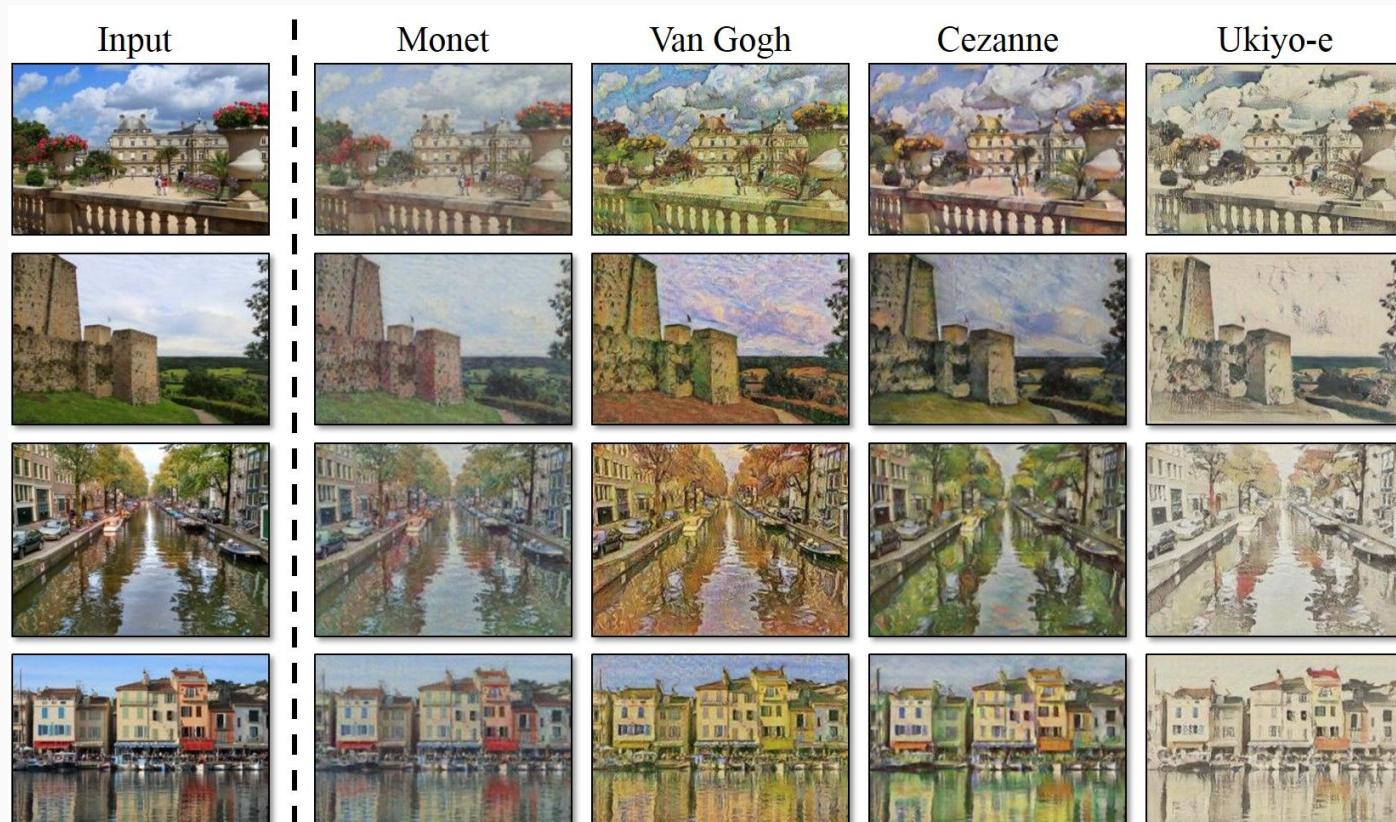


CycleGANs (style transfer)

- GANs de transferência entre domínios provavelmente serão o primeiro lote de aplicativos comerciais.
- Transformam imagens de um domínio para outro domínio.
 - de um cenário real para pinturas de Monet ou Van Gogh
 - de zebras para cavalos (ou vice versa)



Pinte como um mestre



Text to image

- Forneça uma sentença e várias imagens são geradas.



<https://github.com/hanzhanggit/StackGAN>

Face Synthesis

- Síntese de faces em diferentes poses
- Uma única imagem de entrada → faces em diferentes ângulos de visão.
- Uso: transformar imagens para tornar mais fácil o reconhecimento de face.

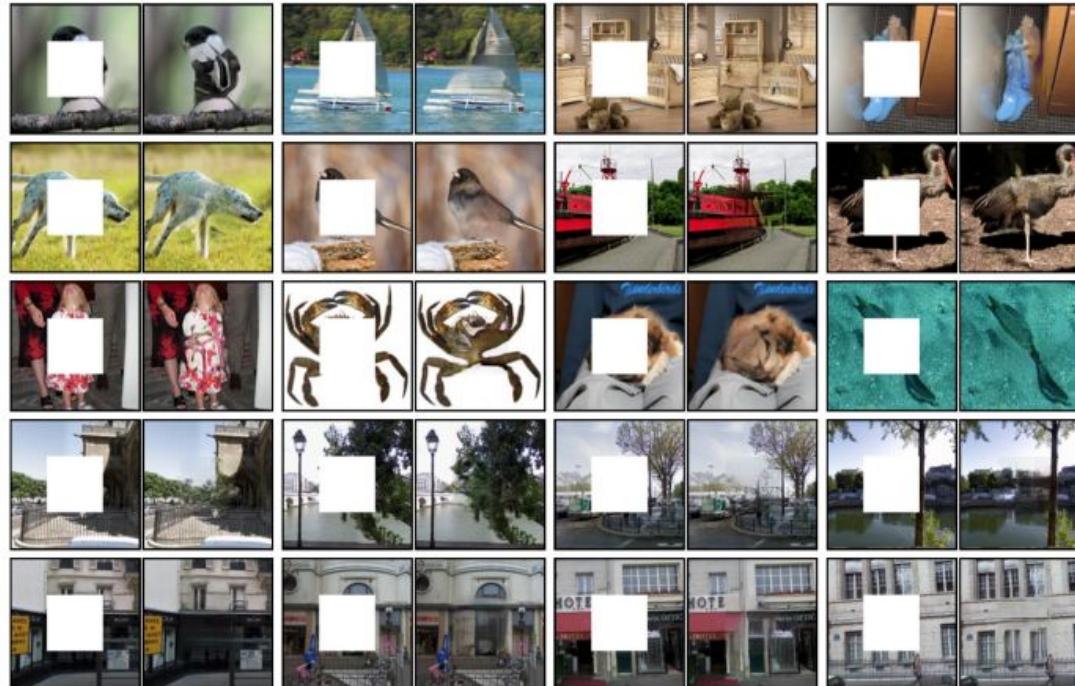


. Synthesis results under various illuminations. The first row is the synthesized image, the second row is the input.

<https://arxiv.org/pdf/1704.04086.pdf>

Image inpainting

- Reparar imagens:
 - preenche a parte que falta com o conteúdo correto.



<https://github.com/pathak22/context-encoder>

Aprender a distribuição conjunta: CoGAN

- É caro criar GANs com diferentes combinações de features faciais:
 - $P(\text{louro, feminino, sorridente, com óculos})$
 - $P(\text{marrom, masculino, sorridente, sem óculos})$
- A maldição da dimensionalidade faz com que o número de GANs cresça exponencialmente.
- Solução: aprender a distribuição INDIVIDUAL das features e combiná-los como se fossem INDEPENDENTES para formar distribuições diferentes.

CoGAN learns the joint probability $P(x_1, x_2)$ by sampling
 $x_1 \sim P(x_1)$ and $x_2 \sim P(x_2)$

- CoGAN : Coupled GAN

<https://arxiv.org/pdf/1606.07536.pdf>

make it blonde

add smile

add glasses



Figure 4: Generation of face images with different attributes using CoGAN. From top to bottom, the figure shows pair face generation results for the blond-hair, smiling, and eyeglasses attributes. For each pair, the 1st row contains faces with the attribute, while the 2nd row contains corresponding faces without the attribute.

DiscoGAN

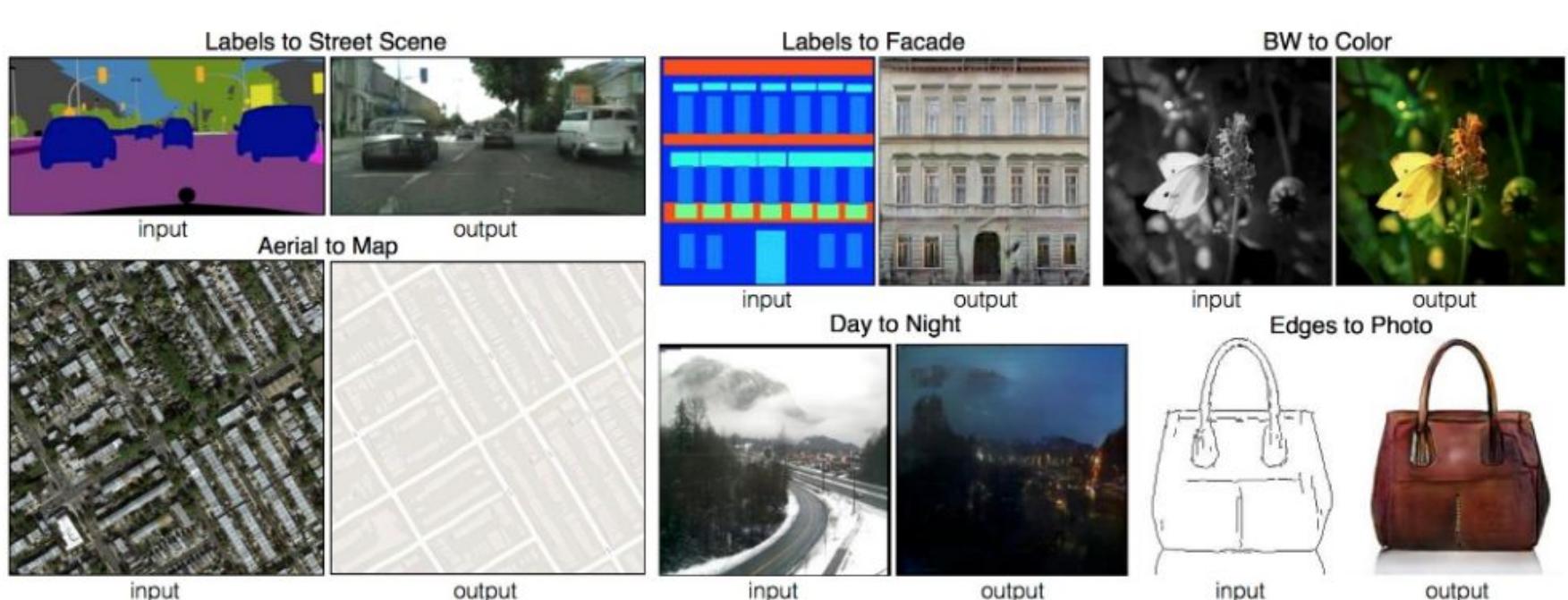
- DiscoGAN fornece matching style.
- Aprende a relacionar domínios diferentes sem rótulos ou pareamentos.
- Por exemplo, transfere o estilo de bolsas para o de sapatos.



<https://github.com/carpedm20/DiscoGAN-pytorch>

Pix2Pix

- "Tradução" de imagem para imagem entre domínios diferentes.



Example results on several image-to-image translation problems. In each case we use the same architecture and objective, simply training on different data.

<https://github.com/phillipi/pix2pix>

- Criar emojis a partir de fotos
- <https://arxiv.org/pdf/1611.02200.pdf>

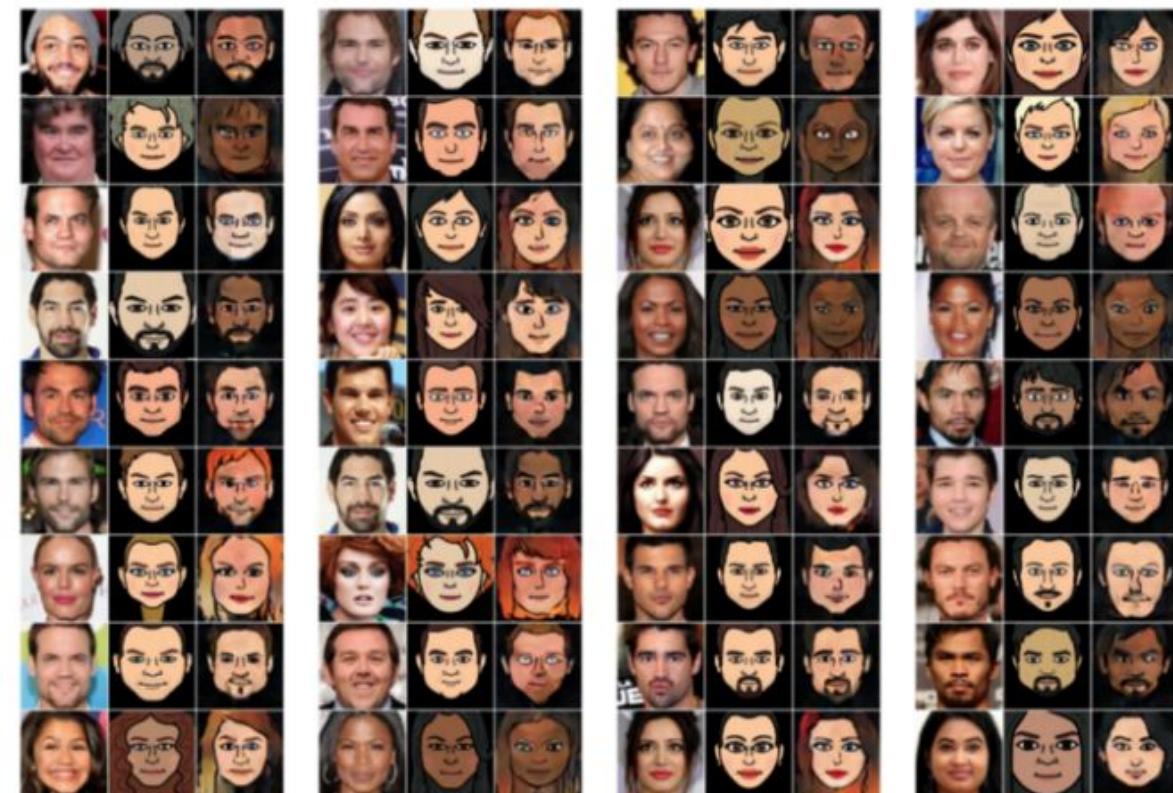


Figure 4: Shown, side by side are sample images from the CelebA dataset, the emoji images created manually using a web interface (for validation only), and the result of the unsupervised DTN. See Tab. 4 for retrieval performance.

Face aging

Face Aging

0-18

19-29

30-39

40-49

50-59

60+



<https://arxiv.org/pdf/1702.01983.pdf>

Geração de músicas

- Pinta e toca e canta...
- <https://deepjazz.io/>

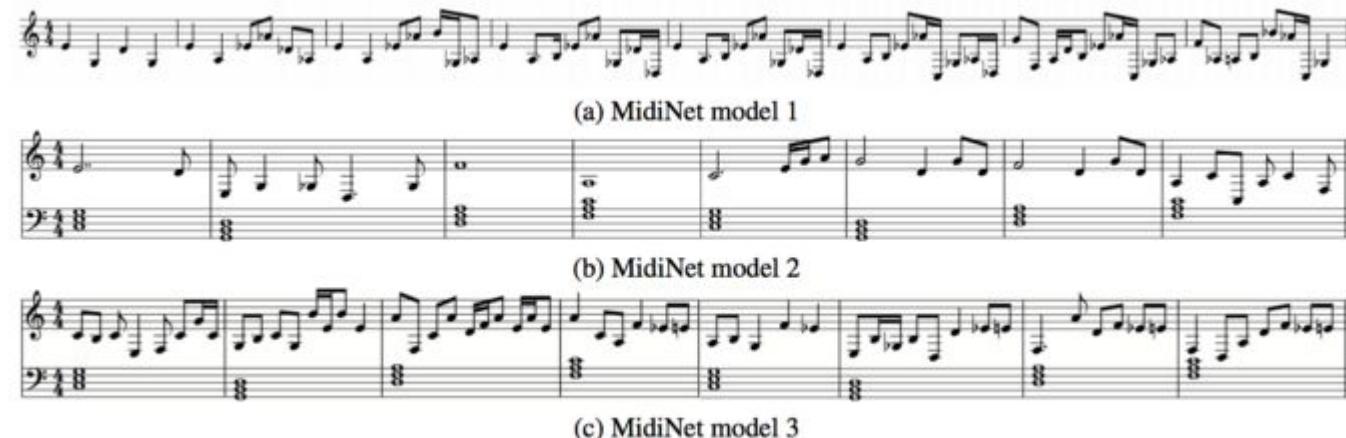


Figure 3. Example result of the melodies (of 8 bars) generated by different implementations of MidiNet.

Como isto funciona? Intuição

- O GAN pode ser visto como um jogo entre dois jogadores:
 - o gerador (generator)
 - e o discriminador (discriminator).
- Gerador:
 - encarregado de gerar novos exemplos, o mais similar possível mas, ao mesmo tempo, diferentes dos exemplos já conhecidos na base de treino.
- Discriminador:
 - oponente do gerador
 - encarregado de detectar que um exemplo é "fake", criado pelo gerador.
- Todos os exemplos anteriores são baseados nesta ideia, que tem sua raiz em teoria dos jogos.

Conceito de Teoria dos Jogos

Teoria dos jogos

- Ramo da matemática aplicada.
- Estuda situações de estratégia:
 - jogadores jogam um contra o outro
 - cada um escolhe uma dentre diferentes ações
 - cada um deles busca minimizar sua perda
 - Ponto fundamental:
 - os jogadores são agentes racionais
 - Um jogador não joga contra a natureza imparcial e indiferente.
 - Cada jogador sabe que o outro é um adversário inteligente
 - Adversário: ele joga contra você; ele procura ganhar; se ele ganhar, você perde.
 - Inteligente: ele sabe que você sabe que ele sabe que você sabe que ele é inteligente
 - Os dois possuem este conhecimento

Exemplo clássico: dilema do prisioneiro

- Dois bandidos presos: A e B
- Existe evidência para condenar cada um deles a 5 anos de cadeia.
- Sabe-se que eles possuem mais informação que pode levar a condenações mais severas, prisão de outros, recuperação do dinheiro roubado, etc.
- Promotoria oferece um acordo de delação premiada:
 - se A delatar, ele sai em 1 ano (e B pega 25 anos)
 - se B delatar, ele sai em 1 ano (e A pega 25 anos)
 - CATCH: se ambos, A e B, delatarem, cada um deles pega 10 anos.
- Qual a melhor estratégia?
 - Cenário 1: eles podem se comunicar
 - Cenário 2: a máfia externa está vigilante
 - Cenário 3: eles não podem se comunicar e não existe máfia

Dilema do prisioneiro no cenário 3

- Tabela de pay-off

		B não delata	B delata
A não delata	5	5	25
A delata	1	25	10

- Qual a melhor estratégia?
- O que quer dizer a "melhor" estratégia?
 - Eliminamos considerações psicológicas, religiosas, morais.
 - O objetivo de cada indivíduo é simplesmente sair o mais rapidamente possível, sem consideração pela sorte do outro.

A estratégia ótima no dilema do prisioneiro

- Tabela de pay-off

		B não delata	B delata
A não delata	5	5	25
A delata	1	25	10

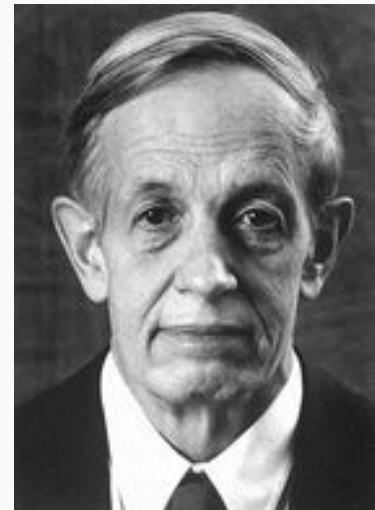
- A melhor estratégia é péssima. Para ambos!!
- Suponha que você é A e considere cada situação possível:
 - B não delata → neste caso, o melhor para A é delatar e sair em 1 ano
 - B delata → neste caso, o melhor para A é delatar (de novo) e sair em 10 anos
- Sob qualquer cenário, o melhor para A é delatar
- Mas B é racional e vai fazer o mesmo raciocínio que A:
 - Ele também vai delatar e ambos pegarão 10 anos quando poderiam pegar 5 anos cada um

Dilema do prisioneiro em todo lugar

- Trânsito complicado:
 - fecha o cruzamento ou não?
 - na estrada para Cabo Frio na véspera de Ano Novo, passar pelo acostamento ou não?
- No jogo de futebol:
 - torcedor da frente se levanta e fica em pé
 - os de trás, também se levantam
 - um efeito cascata e logo todos estão assistindo o jogo de pé
- Dilema é gerado pela função que busca ser minimizada:
 - a minha perda, sem considerar a perda alheia
 - se a função de perda fosse a SOMA das duas perdas individuais, o melhor seria ninguém delatar.

Um pouco de história

- E lá vem ele:
 - John von Neumann
- Publicação em 1944 de *The Theory of Games and Economic Behavior* de John von Neumann e Oskar Morgenstern.
- Outro nome fundamental:
 - John Nash, Nobel em 1994 (personagem do filme Uma Mente Brilhante)
 - Equilíbrio de Nash: nenhum jogador vai lucrar se mudar sua estratégia unilateralmente.



GAN como um jogo

GAN como um jogo

- Dado de cada exemplo: vetor aleatório \mathbf{X}
- Por exemplo, uma imagem 1024x1024 significa um vetor de dimensão 10^6
- Supomos que os exemplos são vetores independentes e identicamente distribuídos com densidade de probabilidade $p(\mathbf{x})$
- Esta distribuição é desconhecida e multi-dimensional
- Ela não varia ao longo do jogo e não depende das ações dos jogadores
- Jogo com dois jogadores:
 - gerador G
 - discriminador D.
- Jogo sequencial: G age, D age, G age, D age, G age, etc....

GAN como um jogo: gerador

- Gerador G:
 - é um modelo generativo de dados: gera exemplos fictícios
 - produz dados $\hat{\mathbf{x}}$ a partir de ruídos aleatórios \mathbf{z}
 - Objetivo do gerador é produzir exemplos $\hat{\mathbf{x}} = G(\mathbf{z})$ com a mesma distribuição $p(\mathbf{x})$ de probabilidade que os dados reais
 - O sonho de consumo do gerador é ser capaz de produzir exemplos $\hat{\mathbf{x}} = G(\mathbf{z})$ que sejam indistinguíveis dos exemplos reais.
 - Em cada momento que deve jogar, o gerador produz um valor $\hat{\mathbf{x}}$
 - A medida que o jogo se desenvolve, o gerador G vai mudando, adaptando-se e melhorando
 - Como isto ocorre exatamente? Veremos daqui a pouco.

GAN como um jogo: a dinâmica

- Gerador e discriminador alternam-se jogando cada um por seu turno
- Na sua vez, gerador produz um exemplo fake $\hat{\mathbf{X}} = G(\mathbf{Z})$
- Quando for a vez do discriminador D jogar, ele é apresentado a:
 - um exemplo real $\mathbf{X} \sim p(\mathbf{x})$
 - OU
 - a um exemplo fake $\hat{\mathbf{X}} = G(\mathbf{Z})$ gerado por G.
- Quem é apresentado, o exemplo real ou o exemplo fake?
 - Isto é decidido ao acaso, jogando-se uma moeda honesta (probab de ser fake = $\frac{1}{2}$).
 - A "moeda" é lançada independentemente a cada rodada do jogo.

GAN como um jogo: discriminador

- Discriminador D:
 - é o outro jogador:
 - oponente do jogador gerador
 - Ele é apresentado a um vetor \mathbf{x} que pode ser real ou fake (foi decidido pela "moeda")
 - D não sabe se o vetor \mathbf{x} ao qual ele é apresentado é real ou fake.
- A tarefa de D é DESCOBRIR se o vetor à sua frente naquele passo é real ou fake.
- Como ele faz isto?
 - Ele possui uma função $D(\mathbf{x}) = \mathbb{P}(\mathbf{x} \text{ é real})$
 - Se $D(\mathbf{x}) > \frac{1}{2} \rightarrow \text{real.}$ Se $D(\mathbf{x}) < \frac{1}{2} \rightarrow \text{fake}$
 - Seu objetivo é que $D(\mathbf{x}) \approx 1$ quando exemplo é real e $D(\mathbf{x}) \approx 0$ quando fake.

O jogo e algumas perguntas

- O gerador será treinado para enganar o discriminador
- Eles podem ser vistos como adversários no jogo.
- Por um lado, para ter sucesso no jogo, o gerador deve fazer amostras que são indistinguíveis para o discriminador.
- Por outro lado, para ter sucesso no jogo, o discriminador deve ser capaz de distinguir as amostras reais das amostras fake.
- As primeiras perguntas técnicas fundamentais são:
 - como obter a função $G(z)$ que o gerador usa para gerar os exemplos fake a partir de ruídos z ?
 - como obter a função $D(x)$ para discriminar entre exemplos reais e fake?
- A seguir, precisamos responder à pergunta:
 - Isto funciona? Por quê?

A função G(z)

Gerando X a partir de ruído Z

- O vetor X possui uma distribuição de probabilidade complicada.
- Imagine que você receba a tarefa de gerar ao acaso um vetor-imagem de um rosto
- Não deve ser um rosto já existente, nem deve "colar" pedaços de rostos distintos (estilo Frankenstein)
- E você deve fazer isto com ruídos aleatórios gerados ao acaso e colocados num vetor z
- Como isto é possível?
- Método da transformada inversa: primeiro método Monte Carlo
- Criação de Stanislaw Ulam (polonês-hoje Ucrânia) e ... John von Neumann (húngaro) em 1947
- Tem certidão de nascimento!

Certidão de nascimento do método Monte Carlo - 1947

THE INSTITUTE FOR ADVANCED STUDY
SCHOOL OF MATHEMATICS
PRINCETON, NEW JERSEY

May 21, 1947

Mr. Stan Ulam
Post Office Box 1663
Santa Fe
New Mexico

Dear Stan,

Thanks for your letter of the 19th. I need not tell you that Klari and I are looking forward to the trip and visit at Los Alamos this Summer. I have already received the necessary papers from Carson Mark. I filled out and returned mine yesterday; Klari's will follow today.

I am very glad that preparations for the random numbers work are to begin soon. In this connection, I would like to mention this: Assume that you have several random number distributions, each equidistributed in $0, 1 : (x^i), (y^i), (z^i), \dots$. Assume that you want one with the distribution function (density) $f(\xi) d\xi : (\xi^i)$. One way to form it is to form the cumulative distribution function: $g(\xi) = \int_0^\xi f(\xi') d\xi'$ to invert it $\xi(x) = \xi \Leftrightarrow x = g(\xi)$, and to form $\xi^i = \xi(x^i)$ with this $\xi(x)$, or some approximate polynomial. This is, as I see, the method that you have in mind.

An alternative, which works if ξ and all values of $f(\xi)$ lie in $0, 1$, is this: Scan pairs x^i, y^i and use or reject x^i, y^i according to whether $y^i \leq f(x^i)$ or not. In the first case, put $\xi^i = x^i$ in the second case form no ξ^i at that step.

The second method may occasionally be better than the first one. In some cases combinations of both may be best; e.g., form random pairs

$$\xi = \sin x, \quad \gamma = \cos x$$

with x equidistributed between 0° and 360° . The obvious way consists of using the sin - cos - tables (with interpolation). This is clearly closely related to the first method. This is an alternative procedure:

$$\text{Put } \xi = \frac{2t}{1+t^2}, \quad \gamma = \frac{1-t^2}{1+t^2}, \quad t = \operatorname{tg} y,$$

with y (which is $\frac{x}{2}$) equidistributed between 0° and 180° . Restrict y to 0° to 45° . Then the ξ, γ will have to be replaced randomly by γ, ξ and again by $\pm \xi, \pm \gamma$. This can be done by using random digits 0, . . . , 7. It is also feasible with

random digits 0, . . . , 9:

0	Replace ξ, γ by	ξ, γ
1	"	$-\xi, \gamma$
2	"	$\xi, -\gamma$
3	"	$-\xi, -\gamma$
4	"	γ, ξ
5	"	$\gamma, -\xi$
6	"	$-\gamma, \xi$
7	"	$-\gamma, -\xi$
8	Reject this digit	
9	"	"

Now $t = \operatorname{tg} y$, $0^\circ \leq y \leq 45^\circ$, lies between 0 and 1, and its distribution function is $\frac{\operatorname{ctg} y}{1+t^2}$. Hence one may pick pairs of numbers t, s both (independently) equidistributed between 0 and 1, and then

use t } for $(1+t^2)s \leq 1$
reject t, s and }
form no t at } for $(1+t^2)s > 1$
this step }

Of course, the first part requires a divider, but the method may still be worth keeping in mind, especially when the ENIAC is available.

* * *

With best regards from house to house.

Yours, as ever,

John

John von Neumann

JvN:MN

JOURNAL OF THE AMERICAN STATISTICAL ASSOCIATION

Number 247

SEPTEMBER 1949

Volume 44

THE MONTE CARLO METHOD

NICHOLAS METROPOLIS AND S. ULAM

Los Alamos Laboratory

We shall present here the motivation and a general description of a method dealing with a class of problems in mathematical physics. The method is, essentially, a statistical approach to the study of differential equations, or more generally, of integro-differential equations that occur in various branches of the natural sciences.

Stanislaw Ulam, 1909 - 1984



Participou do Projeto Manhattan.

Iniciou o projeto Teller-Ulam de armas termonucleares.

Descobriu o conceito de autômato celular

Inventou o método Monte Carlo.
Em 1947, propôs uma abordagem estatística para o problema da difusão de nêutrons em material físsil.

Sugeriu a propulsão de pulsos nucleares.

Em matemática, ele provou vários teoremas e propôs conjecturas.

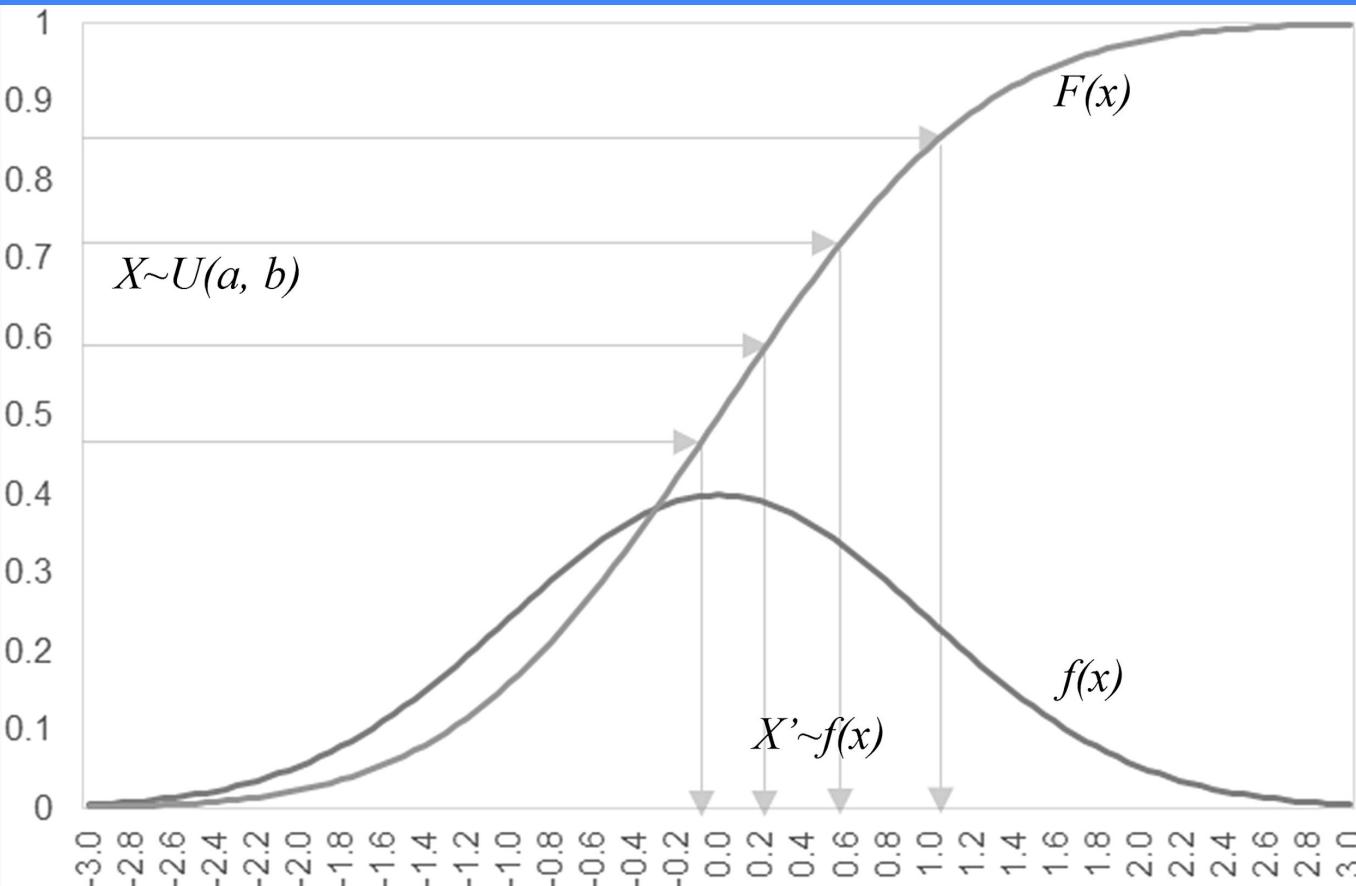


with MANIAC

The lost café - Gian Carlo Rota

<https://fas.org/sgp/othergov/doe/lanl/pubs/00285736.pdf>

Método da transformada inversa



"Veja" a probab de obter $X > 2.0$

ou obter $X < -2.0$

Qual a região onde a maior parte dos dados simulados cairão?

Versão multivariada

- Suponha vetor aleatório X multi-dimensional com densidade de probabilidade $p(x)$
- É possível gerar exemplos seguindo a distribuição de X a partir de vetor aleatório Z i.i.d $U(0,1)$?
- Sim, é possível.
- Obtenha a função de distribuição acumulada $F(x_1, x_2, \dots, x_n)$ da densidade $p(x)$ decomposta em produto de condicionais.
- Gere U_1, U_2, \dots, U_n i.i.d. $U(0,1)$. Inverta usando as condicionais.

Gerando um vetor X com uma distribuição alvo a partir de ruídos i.i.d.

- Podemos escrever a distribuição acumulada multivariada como produto de funções acumuladas condicionais:

$$F_{X_1, X_2, \dots, X_d}(x_1, x_2, \dots, x_d) = F_{X_1}(x_1)F_{X_2|X_1}(x_2|x_1)\dots F_{X_d|X_1X_2\dots X_{d-1}}(x_d|x_1, x_2, \dots, x_{d-1})$$

- Se gerarmos d variáveis aleatórias i.i.d. $U(0,1)$ u_1, u_2, \dots, u_d geramos o vetor X com os passos abaixo:

$$x_1 = F_{X_1}^{-1}(u_1)$$

$$x_2 = F_{X_2|X_1}^{-1}(u_2)$$

⋮

$$x_d = F_{X_d|X_1X_2\dots X_{d-1}}^{-1}(u_d)$$

Mas,..., e daí? O que isto tem a ver com GAN?

- Sabemos que EXISTE uma função G que recebe ruídos Z 's e transforma num vetor X com a densidade desejada
- Esta G é associada com as funções inversas das funções de probabilidade acumuladas.
- Muito difícil de serem obtidas: não sabemos qual é exatamente a distribuição de X
- Solução:
 - imaginar que exista esta G na forma de uma rede neural
 - Assim, o gerador $G(\mathbf{z})$ terá parâmetros $\theta^{(G)}$ (os pesos e biases da rede, e sua arquitetura)
 - Aprender de forma aproximada os parâmetros desta G usando dados.
- Mais sobre isto mais tarde.

A função D

Discriminador

- Ele recebe dois tipos de exemplos: os reais e os fake
 - São duas classes de exemplos: 0 ou 1
 - Como o seu objetivo é classificar corretamente cada exemplo baseado nas features, podemos usar um classificador estatístico para fazer isto.
 - Classificador usado: uma rede neural.
-
- Por exemplo, um conjunto de fotos reais e um conjunto de fotos fake.
 - Se eu tiver os rótulos das fotos, posso usar os pixels como features e criar uma rede neural para classificá-las em reais ou fake.
-
- A rede neural do discriminador é diferente daquela do gerador e possui parâmetros-pesos $\theta^{(D)}$
 - Os pesos-parâmetros controlam a saída da rede discriminador: $D(\mathbf{x}) = \mathbb{P}(\mathbf{x} \text{ é real})$

O jogo é dinâmico

- De forma intuitiva e INFORMAL, o jogo pode ser pensado assim:
- O discriminador vai aprendendo à medida em que tenta acertar a classe.
- Ele recebe um exemplo (ou um mini-batch)
- Discriminador avalia $D(x)$ para cada exemplo e classificar (fake x real)
- Isto é feito com a rede neural $D(x)$ que ele possui no momento.
- Em seguida, a verdadeira classe do exemplo (ou do mini-batch) é revelada.
- Usando stochastic gradient descent ele atualiza seus pesos se tiver errado.

O jogo é dinâmico - 2

- AO MESMO TEMPO, o gerador sabe do resultado alcançado pelo discriminador no mini-batch de fakes and real exemplos que ele (discriminador) recebeu .
- Se o discriminador ficar bom em detectar os exemplos fake, o gerador vai procurar atualizar/melhorar seus pesos.
- O gerador recebe os exemplos reais e fake do mini-batch bem como as classes verdadeiras de cada um e a classificação que o discriminador exerceu.
- Ele pode usar back-propagation para escolher novos coeficientes de ****sua**** rede neural para minimizar o ****seu**** custo.
- O jogo vai sendo jogado até certo equilíbrio ser alcançado (equilíbrio de Nash)
- Esta é uma descrição INFORMAL e vaga. Vamos ver agora de forma mais precisa como isto é feito.

Função de perda do gerador

Jogo de soma zero

- A função de perda do discriminador é

$$J^{(D)}(\theta) = -\frac{1}{2} \mathbb{E}_{p_{\text{real}}} [\log(D(\mathbf{X}))] - \frac{1}{2} \mathbb{E}_{\mathbf{z}} [\log((1 - D(G(\mathbf{z})))]$$

- O tipo de jogo mais simples para se analisar é um jogo de soma-zero.
 - Jogo em que a perda de um jogador é o ganho do outro.
 - A soma das perdas dos dois jogadores é sempre zero.
- Neste caso, tendo a função de perda de D, a função de perda do gerador fica automaticamente determinada:

$$J^{(G)}(\theta^{(G)}, \theta^{(D)}) = -J^{(D)}(\theta^{(G)}, \theta^{(D)})$$

- A “solução” de um jogo de soma-zero é chamada de solução minimax.
- Cada jogador deve escolher a SUA estratégia minimax: isto é o melhor que ele pode almejar contra um jogador racional.

Simplificando um pouco

- A função de perda do gerador é o negativo da perda do discriminador

$$J^{(G)}(\theta^{(G)}, \theta^{(D)}) = -J^{(D)}(\theta^{(G)}, \theta^{(D)})$$

- G quer achar os seus pesos $\theta^{(G)}$ que minimizem a sua perda dada por:

$$J^{(G)}(\theta^{(G)}, \theta^{(D)}) = \frac{1}{2} \mathbb{E}_{p_{\text{data}}} [\log(D(\mathbf{X}))] + \frac{1}{2} \mathbb{E}_{\mathbf{z}} [\log((1 - D(G(\mathbf{z})))]$$

- Note que $\theta^{(G)}$ só aparece na segunda parcela do termo acima.
- Portanto, podemos IGNORAR a primeira parcela ao procurar os pesos ótimos do gerador.
- Mais ainda, estaremos otimizando usando apenas os dados Z, sem olhar para os dados x gerados por p_data(x)
- Isto será relevante no implementação do código da GAN.

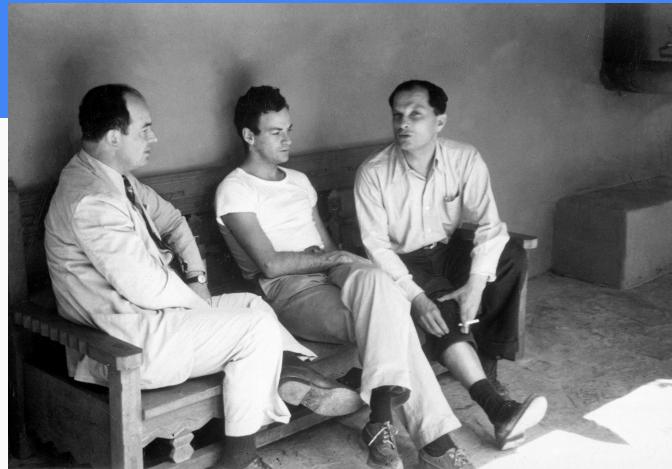
Teorema Minimax

Estratégia minimax em GANs

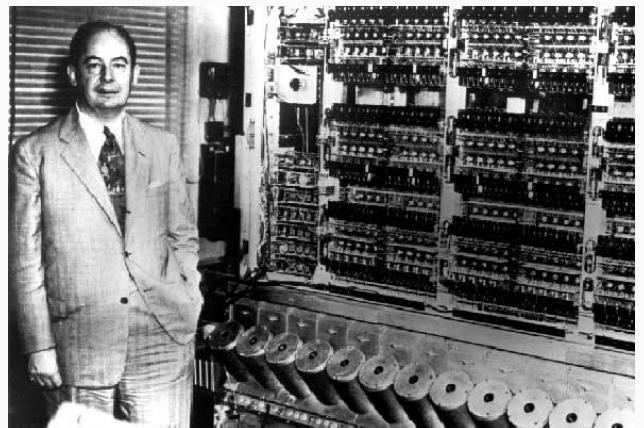
- A solução de um jogo de soma-zero é chamada de solução minimax:
 - cada jogador deve escolher a SUA estratégia minimax.
- Por exemplo, como seria a solução minimax para o jogador G?
- As estratégias de G e de D estão representadas pelos parâmetros $\theta^{(G)}$ e $\theta^{(D)}$
- Para cada estratégia possível para G (isto é, para cada valor de $\theta^{(G)}$):
 - percorra todas as estratégias possíveis de D (representadas por $\theta^{(D)}$)
 - Ache aquela estratégia $\theta_*^{(D)}$ que leva ao pior resultado para G.
 - Isto é, $\theta_*^{(D)} = \max_{\theta^{(D)}} \{ J^{(G)}(\theta^{(G)}, \theta^{(D)}) \}$

Solução minimax: John von Neumann

- Isto tem alguma chance de convergir?
- Tem uma solução?
- Tem uma solução ótima para pelo menos um dos jogadores?
- Uma solução ótima para os dois jogadores?
- Uma solução geral, que valha para todo e qualquer jogo??
- John von Neumann, em 1928:
 - forneceu a resposta para jogos de soma-zero com 2 jogadores.
- Existe solução ótima para os dois jogadores:
 - o melhor que cada um pode almejar é adotar a sua estratégia minimax!



von Neumann, Feynman, Stan Ulam
em Los Alamos,



Solução minimax: John von Neumann

- Solução minimax do jogador G:
 - obteve $\theta_*^{(D)} = \max_{\theta^{(D)}} \{ J^{(G)}(\theta^{(G)}, \theta^{(D)}) \}$
 - Com isto, sabe qual a pior perda que ele pode ter para cada estratégia $\theta^{(G)}$ que adotar
 - Esta pior perda para G quando ela adota a estratégia $\theta^{(G)}$ é $J^{(G)}(\theta^{(G)}, \theta^{(D)*})$
 - G agora vai buscar a estratégia $\theta^{(G)}$ que minimiza a sua pior perda possível.
 - Isto é, sua solução minimax é a estratégia
- Similarmente, o jogador D (se for racional) vai escolher a SUA estratégia minimax

$$\theta_{\text{ótima}}^{(D)} = \min_{\theta^{(D)}} \max_{\theta^{(G)}} \{ J^{(D)}(\theta^{(G)}, \theta^{(D)}) \}$$

Solução minimax de John von Neumann

- Mostra-se que esta solução é um equilíbrio de Nash:
 - se seu adversário é racional, ele sabe fazer análises e vai procurar se proteger contra você
 - Assim, ele vai escolher a solução minimax dele.
 - Sendo assim, você tem de escolher a SUA solução minimax
 - Com os dois jogadores nas sua soluções minimax, nenhum deles tem incentivo para mudar.
 - Se um deles mudar, o outro jogador poderá explorar esta mudança e ganhar mais.
 - O melhor para cada um deles é a solução minimax.
 - Esta solução é um equilíbrio de Nash.

De volta às GANs

- No jogo de soma zero da GAN, a solução ótima do jogo (que minimiza as duas perdas individuais) é a solução minimax
- Para o jogador G, ela será

$$\min_{\theta^{(G)}} \max_{\theta^{(D)}} J^{(G)}(\theta^{(G)}, \theta^{(D)}) = \min_{\theta^{(G)}} \max_{\theta^{(D)}} \left[\frac{1}{2} \mathbb{E}_{p_{\text{data}}} (\log(D(\mathbf{X}))) + \frac{1}{2} \mathbb{E}_{\mathbf{z}} (\log((1 - D(G(\mathbf{z})))) \right]$$

- Similarmente, temos uma expressão para D.
- Na prática este jogo (escolha dos parâmetros das duas redes) é implementado de forma numérica e iterativa.
- **** Esperamos **** (sem garantias) que este procedimento iterativo leve ao equilíbrio de Nash.

Procedimento iterativo de ajuste de uma GAN

- Começamos com valores iniciais $(\theta^{(G)})^{(0)}$ e $(\theta^{(D)})^{(0)}$
- Procedimento envolve dois passos alternados:
 - 1) fixamos $\theta^{(G)}$ no seu valor corrente e atualizamos $\theta^{(D)}$
 - 2) fixamos $\theta^{(D)}$ no seu valor corrente e atualizamos $\theta^{(G)}$
 - Com sorte, este procedimento converge e com muita sorte, ele converge para o equilíbrio de Nash
- As atualizações são feitas usando mini-batches e passos de stochastic gradiente descendente
$$\theta^{(D)} \leftarrow \arg \max_{\theta^{(D)}} \left[\frac{1}{2} \mathbb{E}_{p_{\text{real}}} (\log(D(\mathbf{X}))) + \frac{1}{2} \mathbb{E}_{\mathbf{z}} (\log((1 - D(G(\mathbf{z})))) \right]$$
 - Rótulos com a decisão de D no mini-batch são obtidos on-the-fly usando o $\theta^{(D)}$ corrente

Procedimento iterativo de ajuste de uma GAN

- Similarmente, G busca seu mínimo num passo SGD para $\theta^{(D)}$ fixo

$$\theta^{(G)} \leftarrow \arg \min_{\theta^{(G)}} \left[\underbrace{\frac{1}{2} \mathbb{E}_{p_{\text{real}}} (\log(D(\mathbf{X}))) + \frac{1}{2} \mathbb{E}_{\mathbf{z}} (\log((1 - D(G(\mathbf{z}))))}_{\text{nao envolve } \theta^{(G)}} \right]$$

- Assim, podemos atualizar os pesos do gerador sem usar os dados x da distribuição p_real:

$$\theta^{(G)} \leftarrow \arg \min_{\theta^{(G)}} \left[\frac{1}{2} \mathbb{E}_{\mathbf{z}} (\log((1 - D(G(\mathbf{z})))) \right]$$

- Você verá que, no passo para atualizar G, vamos usar apenas os dados fake.

Detalhes do treinamento

- Na prática, não temos as esperanças nas fórmulas anteriores.
- Substituímos as expressões das esperanças por médias aritméticas baseadas em mini-batches.
- Amostre m ruídos gaussianos $\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \dots, \mathbf{z}^{(m)}$ da densidade a priori $p(\mathbf{z})$
- Pegue um mini-batch de tamanho m dos exemplos reais: $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(m)}$
- Passo SGD usando ADAM ou outro minimizador:

$$\theta^{(D)} \leftarrow \theta^{(D)} + \varepsilon \nabla_{\theta^{(D)}} \frac{1}{m} \sum_{i=1}^m \left[\log(\underbrace{D}_{\theta^{(D)} \text{ aqui}}(\mathbf{x}^{(i)})) + \log(1 - \underbrace{D}_{\theta^{(D)} \text{ aqui}}(G(\mathbf{z}^{(i)}))) \right]$$
$$\theta^{(D)} \leftarrow \theta^{(D)} + \varepsilon \frac{1}{m} \sum_{i=1}^m \left[\frac{1}{D(\mathbf{x}^{(i)})} \nabla_{\theta^{(D)}} D(\mathbf{x}^{(i)}) - \frac{1}{1 - D(G(\mathbf{z}^{(i)}))} \nabla_{\theta^{(D)}} D(G(\mathbf{z}^{(i)})) \right]$$

Detalhes do treinamento

- Amostre m ruídos gaussianos $\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \dots, \mathbf{z}^{(m)}$ da densidade a priori $p(\mathbf{z})$
- Pegue um mini-batch de tamanho m dos exemplos reais: $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(m)}$
- Passo SGD para o gerador:

$$\theta^{(G)} \leftarrow \theta^{(G)} - \varepsilon \nabla_{\theta^{(G)}} \frac{1}{m} \sum_{i=1}^m \log(1 - D(\underbrace{G}_{\theta^{(G)} \text{ aqui}}(\mathbf{z}^{(i)})))$$
$$\theta^{(G)} \leftarrow \theta^{(G)} - \varepsilon \frac{1}{m} \sum_{i=1}^m \frac{-1}{1 - D(G(\mathbf{z}^{(i)}))} \nabla_{\theta^{(G)}} D(G(\mathbf{z}^{(i)}))$$

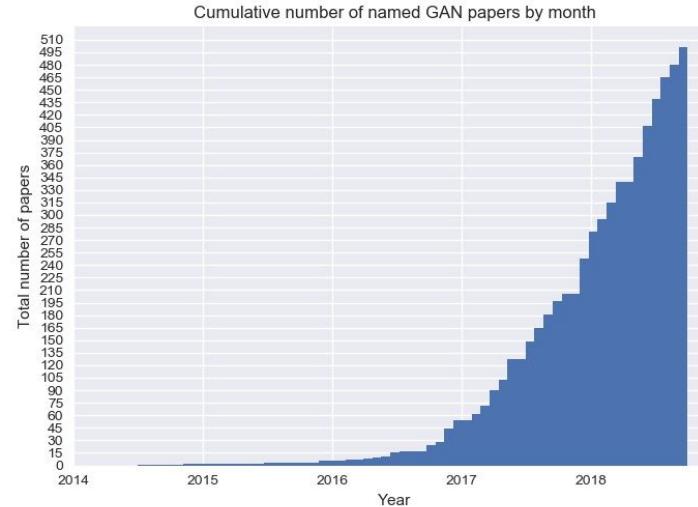
Resumindo e um passo à frente

GANs

- After seeing the vanilla GAN, we will cover different, more specialized, GANs
 - **Standard GAN:** GAN, DCGAN.
 - **Conditional GAN:** cGAN, SS-GAN, InfoGAN, ACGAN.
 - **Different Loss Function:** WGAN, WGAN-GP, LSGAN.
 - **Image Translation:** Pix2Pix, CycleGAN.
 - **Advanced GANs:** BigGAN, PG-GAN, StyleGAN.
 - **Other:** StackGAN, 3DGAN, BEGAN, SRGAN, DiscoGAN, SEGAN.
- We will also cover some additional aspects related to GANs:
 - Evaluating GANs
 - Wasserstein GAN

Um crescimento exponencial de GANs

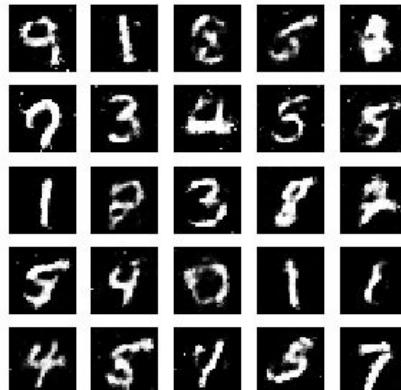
De <https://github.com/hindupuravinash/the-gan-zoo>



Implementação em Keras: <https://github.com/eriklindernoren/Keras-GAN>

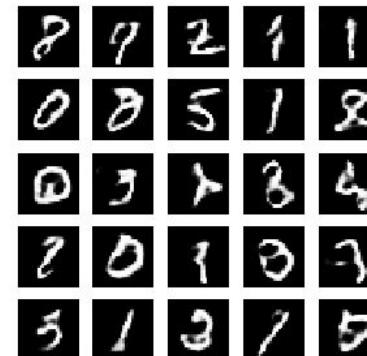
DCGAN

- GAN original: discriminador e o gerador são redes totalmente conectadas
- DCGAN: usa Deep convolutional neural nets
- Ver <https://github.com/vmartinezalvarez/Image-generation-GAN-vs-DCGAN>



Epoch 200

GAN: fully connected

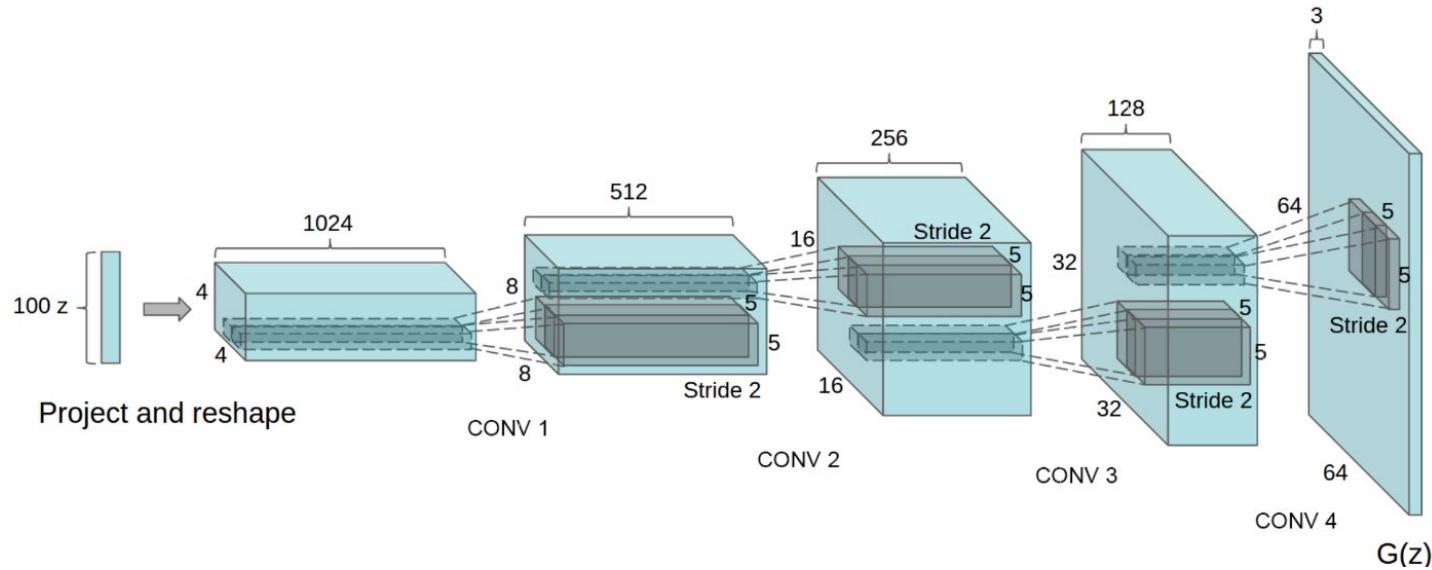


Epoch 100

DCGAN

DCGAN

- Generator architecture:
 - Latent dimension = 100
 - Output = image 64x64 with 3 channels



cGAN - Conditional GAN

cGAN

- **Conditional Generative Adversarial Network (cGAN)**
- Mirza and Osindero, 2014 Arxiv (mesmo ano que paper GAN original)
- Não foi publicado em nenhuma conferência mas possui > 6600 citações
- Nós particionamos o conjunto de entradas (imagens) em subconjuntos.
- Cada subconjunto é identificado por um label discreto y
 - MNIST:
 - subconjunto das imagens dos 0's, subconjunto das imagens de 1's, etc.
 - CIFAR10:
 - 10 diferentes classes de imagens
 - airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks.
 - Faces:
 - masculinas e femininas

Uncontrolled sampling

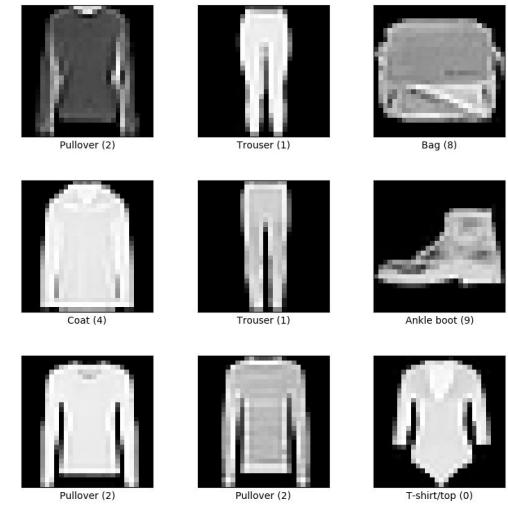


Fashion MNIST - 10 classes

Uncontrolled sampling



Amostra gerada por GAN

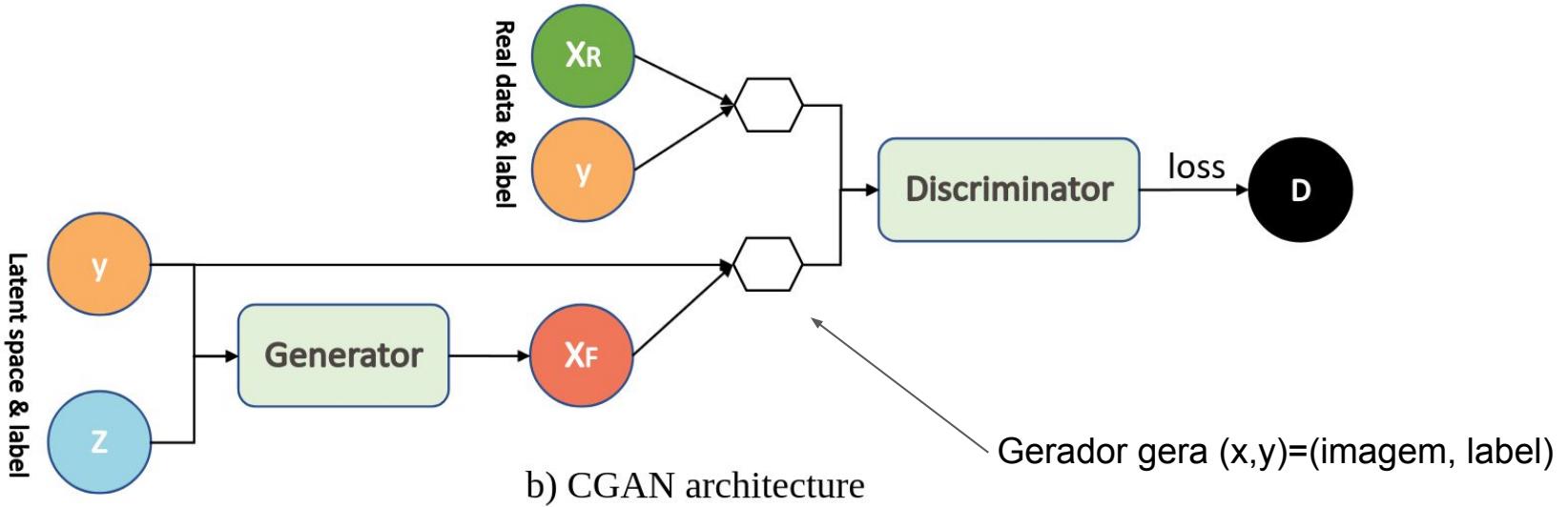
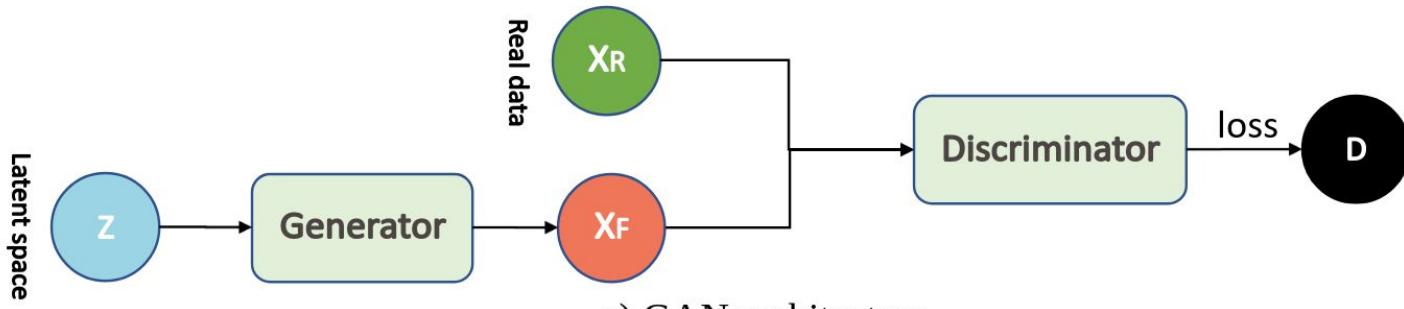


Sem sandálias...
Quantas preciso gerar até
obter uma sandália?

cGAN

- GAN vanilla com MNIST:
 - Dataset usa imagem x (28 x 28)
 - cGAN com MNIST
 - Dataset usa imagem x e o label y: $(x, y) \rightarrow (28 \times 28 \times 1)$
 - A diversidade da população de imagens é quebrada em subpopulações (ou classes) mais homogêneas.
 - Ainda existe diversidade:
 - em MNIST, os 1's ou 2's não são todos iguais
 - existe variação dentro da classe do dígito
 - Mas a maior parte da variabilidade do MNIST deve ser reduzida ao condicionar (fixar) a sua classe
 - As imagens dentro de uma classe são mais parecidas

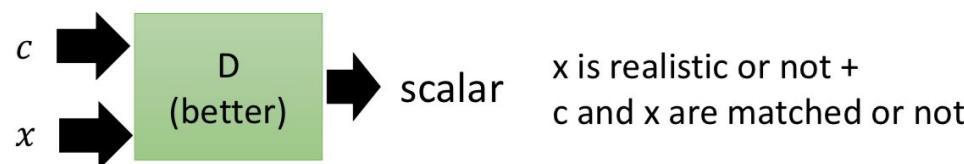
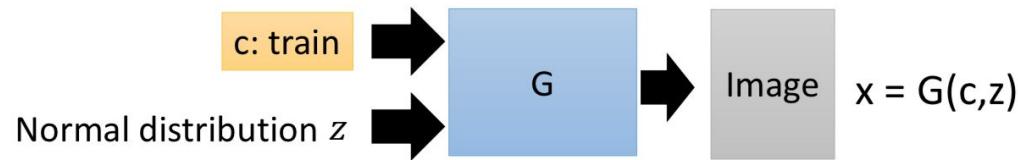




Discriminador precisa checar consistência de label e imagem

[Scott Reed, et al, ICML, 2016]

Conditional GAN



True text-image pairs: (train ,) 1

(cat ,) 0 (train ,) 0

cGAN

- O que muda em cGAN? A função de perda
- GAN usual:

$$\min_G \max_D \mathbb{E}_{x \sim p_{\text{data}(x)}} [\log D(x)] + \mathbb{E}_{z \sim p_{Z(z)}} [\log(1 - D(G(z)))]$$

- cGAN:

$$\min_G \max_D \mathbb{E}_{x \sim p_{\text{data}(x)}} [\log D(x|y)] + \mathbb{E}_{z \sim p_{Z(z)}} [\log(1 - D(G(z|y)))]$$

Mas como implementar isto? Com um truque super-legal no paper original: embedding...

Solução ruim: Que tal criar uma GAN para cada classe?

- Ao passar o label y da classe, criamos uma rede neural para cada diferente valor de y
- O gerador será da forma $x \sim G_y(z)$
 - Haverá um gerador G gerando x quando $y=0$.
 -
 - Isto é, teremos um primeiro gerador $x \sim G_0(z)$
 - Haverá OUTRO gerador G gerando x quando $y=1$. Isto é, $x \sim G_1(z)$
 - Haverá um terceiro gerador $x \sim G_2(z)$
 -
 - Temos um gerador especializado para cada classe y possível
 - Temos também um discriminador especializado naquele gerador-classe

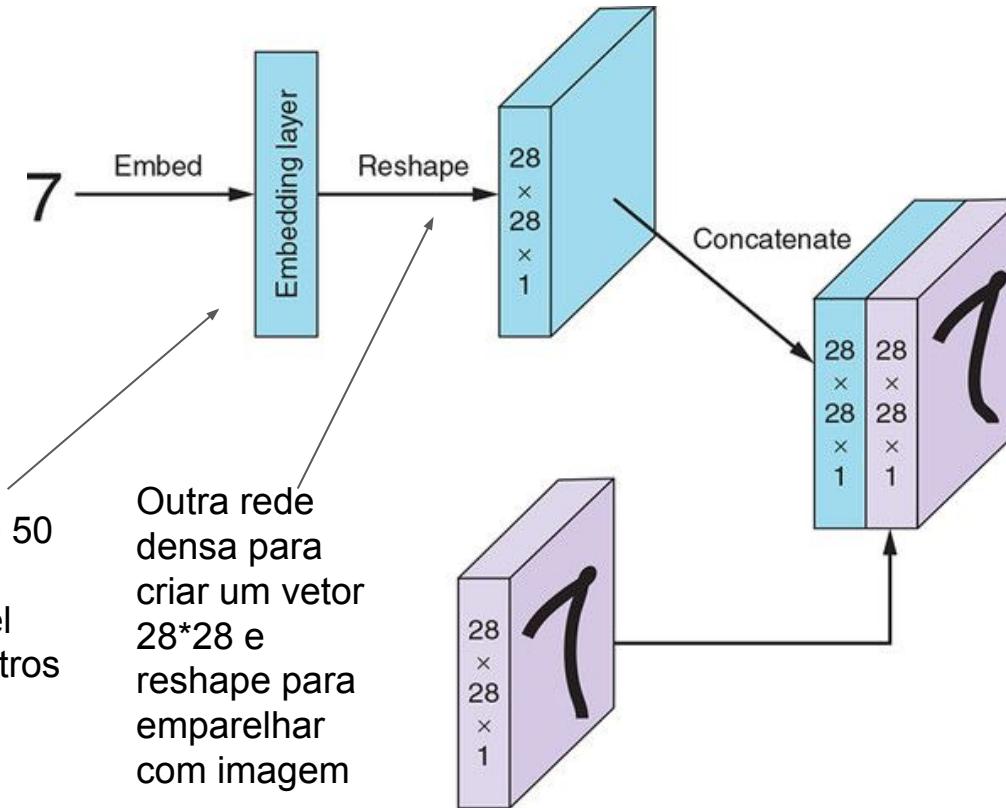
Uma GAN para cada classe?

- Uma GAN ($G + D$) para cada classe: $\#GANs = \#classes$.
- Se $\#classes$ for grande e G e D forem redes complexas, ficará inviável.
- Ao criar GANs independentes para cada classe perdemos a chance de economizar pois várias classes podem ter features parecidas.
- Por exemplo, em MNIST, os dígitos 1's e 9's usam um segmento de reta.
- Poderíamos usar esta redundância sem precisar replicar o aprendizado em cada classe.
- E se quisermos condicionar em algo mais complexo que o rótulo de classe?
- Por exemplo, condicionar num texto? Outra imagem (pix2pix)?
- Solução: criar um embedding para a informação condicionante e misturá-lo com os dados.

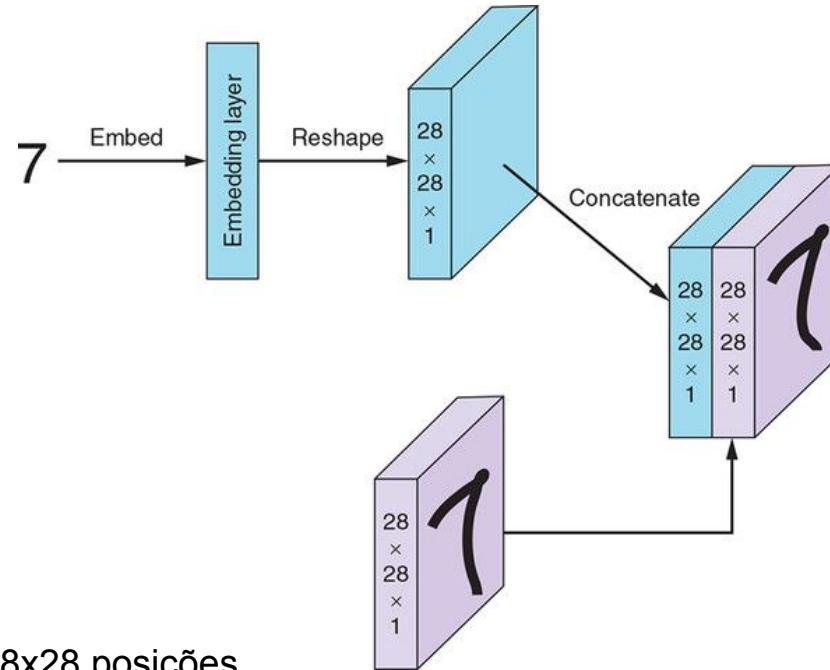
cGAN - embedding the class label

- Truque muito esperto: trocar o rótulo discreto y por um vetor de dimensão “grande” com entradas contínuas
- Em seguida, concatene o vetor contínuo com a imagem de entrada
- Por exemplo, em MNIST:
 - temos 10 classes com $y = 0, 1, 2, \dots, 9$
 - Cada inteiro y será transformado num vetor euclidiano de dimensão 50
 - Dez classes $y \rightarrow$ 10 diferentes vetores de dimensão 50.
 - Parece um desperdício: um número inteiro que requer 4 bits para ser codificado será transformado em um array com 50 floats de double precision!
 - Mas funciona bem: pode ser generalizado para qualquer número de classes: se tivermos 10 classes ou 1000 classes, o embedding com um vetor de dimensão 50 funciona.
 - Além disso, pode ser generalizado para informações adicionais como texto, por exemplo.

MNIST com imagens 28 x 28



MNIST com imagens 28 x 28



Em algumas implementações,
depois de concatenar, cria-se
uma única imagem 28 x 28
multiplicando pixel a pixel os
dois canais.

Outra opção: repete “7” em todas as 28x28 posições.

Ver <https://github.com/abhishekhsuran/myGAN-tf2.0/blob/master/mnistConditionalGAN.ipynb>

