Roteiro da apresentação

Introdução

Bom dia a todos. Nossa apresentação de hoje é sobre Redes Adversariais Generativas. A ideia central é cobrir desde a motivação até as variantes mais avançadas como a StyleGAN, seguindo o sumário apresentado aqui: Introdução, Definição, a Arquitetura, a Base Matemática, os Modelos Derivados e as Considerações Finais

No processo de aprendizado, eu sempre busco explorar as motivações internas sobre as coisas. Muitas vezes elas podem ser respostas simples como "Por que não?". Se analisarmos o histórico de desenvolvimento do aprendizado de máquina, veremos que maioria das soluções práticas de Machine Learning sempre foram discriminativas, ou seja, modelos criados para tarefas de classificação e regressão. Eles foram os que primeiro alcançaram sucesso prático em larga escala e podemos destacar:

- Redes Neurais Convolucionais (CNNs)
- Regressão Logística (Logistic Regression)
- Máquinas de Vetores de Suporte (SVM)
- Árvores de Decisão e Random Forests

E algumas vezes, o sucesso de uma ferramenta, cria a ilusão de que o limite foi alcançado.

Pegando um exemplo emprestado da astronomia, temos o telescópio Hubble. Feito para medir a constante de Hubble (expansão do universo) foi usado para descobrir a energia escura, mapear o campo profundo, medir massa de buracos negros, revelar proto-galáxias, estudar atmosferas de exoplanetas. Um telescópio focado em uma única constante mudou toda a astrofísica. Creio que através de perguntas ou experiências que indicavam que era possível ir além.

Acredito que em aprendizado a pergunta que pode ter mudado o rumo foi: E se por acaso as máquinas pudessem entender e reproduzir a origem dos dados, não apenas a sua separação.

Enquanto nossos modelos discriminativos buscam a fronteira entre as classes respondendo "O que ela é?". Existe a possibilidade de buscar a estrutura interna de cada classe e responder "Como ela foi feita?".

O que modelos generativos fazem de diferente é modelar a probabilidade dos dados $\mathrm{P}(X)$ ou a probabilidade conjunta $\mathrm{P}(X, Y)$. Não se limitando a dizer que "este pixel pertence à classe A". Eles tentam aprender as dependências e correlações complexas que fazem com que um dado X se pareça com a realidade. É uma tentativa de aprender a lei de formação dos dados.

Modelos discriminativos não têm saída para o problema de amostragem. Não há um mecanismo interno que permita gerar um novo X. Eles apenas pegam um X existente e dão retornam um Y. Modelos Generativos depois de aprender a distribuição $\mathrm{P}(X)$, podem ser amostrados.

Imagine que você aprende a distribuição:

- “Como é o formato típico de um rosto humano?”
- “Como são as letras manuscritas?”
- “Como é o som típico de voz?”
- “Como é a estrutura típica de uma frase?”

Depois que o modelo aprende P(X), você pode “pedir a ele”: Me dê um novo X que seja plausível dentro dessa distribuição.

Em essência, a ideia era mudar o foco de "o que é" para "como é feito". A capacidade de geração era apenas a prova de que o modelo realmente entendeu a estrutura geral, não apenas suas classes.

Transformação estatística

Acredito que estamos agora devidamente apresentados as motivações do surgimento desse assunto. Um dos pontos que vamos discutir agora é o fato de que quando as GANs surgiram, já existiam modelos generativos. Então o que mudou?

Agora vamos de um simples ruído à amostragem complexa. Para todos os modelos generativos, o conceito central é a geração de variáveis aleatórias complexas X a partir de uma distribuição conhecida e simples, como uma distribuição uniforme ou gaussiana.

O Desafio da Aleatoriedade

Um computador é fundamentalmente determinístico, mas pode gerar números pseudo-aleatórios. O caso mais simples é uma sequência que segue uma *distribuição uniforme sobre $[0, 1]$. O desafio é transformar essa distribuição uniforme simples em distribuições complexas, como a distribuição de pixels em imagens realistas.

O Método de Transformação Inversa

A ideia do Método de Transformação Inversa é representar uma variável aleatória complexa X como o resultado de uma função de transformação aplicada a uma variável aleatória uniforme U.

1. Função de Distribuição Cumulativa (CDF): Uma variável aleatória X é totalmente definida por sua Função de Distribuição Cumulativa (CDF), $\mathrm{CDF}_X(x)$, definida em uma dimensão como:

   $$CDF_X(x) = \mathbb{P}(X \leq x) \qquad x \in [0, 1]$$

   Para a variável aleatória uniforme U sobre $[0, 1]$, temos:

   $$CDF_U(u) = \mathbb{P}(U \leq u) = u \quad \forall u \in [0, 1]$$

2. A Transformação: Se considerarmos CDF_X inversível e definirmos uma nova variável Y tal que:

   $$Y = CDF_X^{-1}(U)$$

   onde CDF_X^{-1} é a função inversa.

3. Resultado Estatístico: Podemos demonstrar que Y e X possuem o mesmo CDF:

   $$CDF_Y(y) = \mathbb{P}(Y \leq y) = \mathbb{P}(CDF_X^{-1}(U) \leq y) = \mathbb{P}(U \leq CDF_X(y)) = CDF_X(y)$$

Como Y e X têm o mesmo CDF, eles definem a mesma variável aleatória.

Para resumir, esse método usa uma função de transformação (CDF^{-1}) para deformar e remodelar a distribuição de probabilidade inicial, que é uniforme, na distribuição de probabilidade alvo (X).

Onde entram as GANs?

O gerador de uma GAN faz exatamente isso, mas sem precisar conhecer CDF_X nem sua inversa. Em vez de fornecer uma fórmula explícita para a transformação CDF_X^{-1}, o gerador aprende automaticamente uma função altamente complexa:

$G(z) \simeq F^{-1}(z)$

onde: z é um vetor de ruído simples G é uma rede neural totalmente diferenciável $G(z)$ produz amostras com a mesma distribuição dos dados reais

O papel do discriminador é fornecer o sinal de aprendizado necessário para que G descubra, gradiente a gradiente, qual é a transformação correta para mapear ruído simples em dados realistas, algo que seria impossível de calcular analiticamente em espaços de alta dimensão como imagens.

O insight da GAN é substituir a matemática impossível do CDF_X^-1 por uma rede neural que aprende essa transformação diretamente dos dados. Se você ainda não se emocionou, essa é a hora. Brincadeira, mas isso é incrível.

Antes do surgimento das GANs, vários modelos generativos já buscavam resolver o mesmo problema central: gerar amostras realistas de uma distribuição complexa a partir de uma distribuição simples, normalmente um vetor de ruído gaussiano ou uniforme. Porém, cada abordagem tinha limitações técnicas que dificultavam seu uso em escala.

1. Modelos Gráficos Probabilísticos (PGMs)

Ex.: Bayesian networks, Markov Random Fields, Boltzmann Machines.

Esses modelos representavam relações probabilísticas entre variáveis e tentavam modelar explicitamente a distribuição conjunta dos dados.

Limitações

- Exigem calcular ou aproximar distribuições condicionais complexas.
- Escalam mal para dados de alta dimensão, como imagens.
- O processo de amostragem é caro e lento.

2. Restricted Boltzmann Machines (RBMs)

As RBMs tentam aprender a distribuição de probabilidade dos dados usando camadas restritas de nós visíveis e ocultos.

Limitações

- Treinamento baseado em Contrastive Divergence, que é aproximado e instável.
- Dificuldade de aprender dados complexos.
- Pouco escaláveis para grandes redes profundas.

3. Deep Belief Networks (DBNs)

Empilham várias RBMs, formando um modelo generativo profundo.

Limitações

- Treinamento em múltiplas etapas (camada por camada).

- Amostragem lenta.
- Resultados em imagens tendem a ser borrados e limitados.

4. Autoencoders Variacionais (VAEs)

Método probabilístico que aprende a mapear dados para uma distribuição latente (codificador) e depois reconstruí-los (decodificador).

Esse aqui foi desafiador, por ter um treinamento estável e uma base matemática forte, mas...

Limitações

- A função de perda envolve uma regularização que tende a produzir imagens borradas.
- Difícil aprender detalhes de alta frequência (Bordas nítidas, textura fina, transições suaves)

Definição

Proposta por Ian Goodfellow e alguns colaboradores publicado no NIPS 2014 e considerada por Yann Lecun, um dos grande nomes que temos na área de aprendizado de máquina e redes neurais: "A ideia mais interessante nos últimos 10 anos em aprendizado de máquina.

Elas surgiram com uma ideia fundamentalmente diferente:

Elas não tentam modelar a distribuição de probabilidade explicitamente. Em vez disso, tratam a geração como um jogo competitivo entre dois jogadores:

O Gerador aprende a transformar ruído em dados (como seu texto já explica brilhantemente usando o paralelo com o método de transformação inversa).

O Discriminador aprende a distinguir amostras reais das criadas pelo Gerador.

Essa estrutura resolve as limitações dos modelos anteriores:

1. Treinamento direto na amostragem

As GANs aprendem apenas a gerar amostras, sem precisar calcular verossimilhança, integrais ou normalizações.

2. Amostras mais realistas

Por não depender de aproximações como KL divergences do VAE, o Gerador pode focar na nitidez e fidelidade visual, guiado pelo Discriminador.

3. Não precisam do CDF inverso
4. Escalam bem para imagens de alta dimensão

GANs foram o primeiro modelo generativo realmente eficaz para imagens de alta resolução (DCGAN, 2015).

Uma definição formal:

As Redes Adversariais Generativas (GANs), são modelos de aprendizado profundo formados por duas redes neurais que competem entre si: um gerador, responsável por criar novos dados a partir de um ruído aleatório, e um discriminador, que tenta distinguir entre dados reais do conjunto de treinamento e dados falsos

produzidos pelo gerador. As duas redes são treinadas simultaneamente em um jogo de soma zero, no qual o gerador busca enganar o discriminador, enquanto o discriminador tenta não ser enganado. Esse processo competitivo faz com que ambas as redes melhorem continuamente, até que o gerador se torne capaz de produzir dados tão realistas que o discriminador não consiga mais diferenciá-los dos dados originais.

O potencial das GANs está no fato de que podem aprender a imitar qualquer distribuição de dados, pelo menos na teoria.Teoricamente, podem ser ensinadas a criar mundos estranhamente semelhantes aos nossos em qualquer domínio: imagens, música, fala, prosa. Não sei se eu forçaria em dizer que são artistas robóticos ou apenas um neorônio espelho super poderoso.

Vamos entender melhor como isso é feito.

Arquitetura

Não que eu me importe tanto com semântica, mas o termo "adversariais" advém do fato de serem duas redes colocadas uma contra a outra. Vejam só esse exemplo para um conjunto de imagens. Inclusive esse diagrama foi gerado por um modelo.

O funcionamento envolve o treinamento simultâneo desses dois modelos, geralmente definidos por multilayer perceptrons (MLPs), que são chamados redes adversariais.

O gerador recebe como entrada um vetor de ruído amostrado aleatoriamente, normalmente extraído de uma distribuição gaussiana. Esse vetor é processado por uma rede neural — geralmente composta por camadas convolucionais transpostas ou blocos de upsampling — até resultar em uma imagem sintética. Essa imagem é então enviada ao discriminador, uma segunda rede neural cujo objetivo é decidir se a imagem pertence ao conjunto de dados real ou se foi produzida artificialmente pelo gerador.

Como controlamos o processo de treinamento, sabemos exatamente o rótulo de cada imagem apresentada ao discriminador (real ou falsa). Com esses rótulos, calculamos a função de perda e aplicamos backpropagation para aprimorar o discriminador.

O ponto crucial é que o gerador também é totalmente diferenciável. Assim, ao conectar gerador e discriminador em sequência, podemos propagar o gradiente da perda através do discriminador e de volta ao gerador, permitindo que ambos sejam atualizados — cada um a partir do mesmo sinal de aprendizado. O treinamento é efetivo apenas quando há um equilíbrio: o discriminador não pode ser forte demais (senão o gerador não aprende), nem fraco demais (senão o jogo perde sentido).

Se esse processo permanecer estável por tempo suficiente, o gerador evolui continuamente com o feedback do próprio discriminador, tornando-se um “falsificador” altamente competente, capaz de criar dados extremamente semelhantes aos do conjunto original.

Vamos pegar as etapas usadas no artigo que deu origem a essa arquitetura.

1. O gerador considera números aleatórios e retorna uma imagem.
2. Essa imagem gerada é inserida no discriminador ao lado de um fluxo de imagens tiradas do conjunto de dados real e verdadeiro.
3. O discriminador obtém imagens reais e falsas e retorna probabilidades, um número entre 0 e 1, com 1 representando uma previsão de imagem autêntica e 0 representando previsão de imagens falsas (geradas pela rede generativa).

Para o MNIST, a rede discriminadora é uma rede convolucional padrão que pode categorizar as imagens alimentadas, um classificador binomial que rotula as imagens como reais ou falsas. Já o gerador podemos chamar de é uma rede convolucional inversa, em certo sentido: enquanto um classificador convolucional padrão recebe uma imagem e reduz a amostragem para produzir uma probabilidade, o gerador pega um vetor de ruído aleatório e faz o upsample para uma imagem.

O primeiro joga fora os dados por meio de técnicas de downsampling, como o maxpool, e o segundo gera novos dados. Ambas as redes estão tentando otimizar uma função de perda diferente e oposta. À medida que o discriminador muda seu comportamento, o gerador também muda e vice-versa. Suas perdas empurram um contra o outro. Durante o treinamento, a rede generativa vai aprendendo a criar uma imagem fake que fica cada vez mais próxima de uma imagem real.

Creio que isso seja suficiente para iniciar a compreensão geral de como fazer, mas como somos estudantes de pós-graduação, claramente isso não nos satisfaz. Até onde é possível? Bom, isso é uma pergunta mais complexa que esse modelo. Então vamos partir para algumas ferramentas matemáticas, estatística e de programação usadas para construir essa coisa.

Aprofundamento matemático

As redes GAN são formuladas por um problema como um jogo minimax de dois jogadores. O framework é mais simples de aplicar quando ambos os modelos, G e D, são multilayer perceptrons (MLPs).

Definição dos Modelos

1. Gerador (G): O gerador G é uma função diferenciável $G(z;\theta_g)$, representada por um MLP com parâmetros θ_g. Ele transforma uma variável de ruído de entrada z, que é amostrada de uma distribuição anterior conhecida $p_z(z)$, no espaço de dados x. Implicitamente, G define uma distribuição de probabilidade p_g (a distribuição das amostras $G(z)$).
2. Discriminador (D): O discriminador D é um segundo MLP, $D(x;\theta_d)$, que produz um único escalar $D(x)$. $D(x)$ representa a probabilidade de que x tenha vindo da distribuição de dados reais (p_{data}) em vez da distribuição gerada por G (p_g).

O Jogo Minimax

D e G jogam o seguinte jogo minimax de dois jogadores com a função valor $V(G,D)$:

$$\min_G \max_D V(D,G) = E_{x\sim p_{\text{data}}(x)}[\log D(x)] + E_{z\sim p_z(z)}[\log(1-D(G(z)))] \quad (\text{Equação } 1)$$

O objetivo de cada modelo é o seguinte:

- D é treinado para maximizar $V(D,G)$. Isso equivale a maximizar a probabilidade de atribuir o rótulo correto tanto aos exemplos reais (x) quanto aos exemplos falsos ($G(z)$).
- G é treinado simultaneamente para minimizar $V(D,G)$, o que é equivalente a minimizar $\log(1-D(G(z)))$.

Implementação Prática do Treinamento

O sistema inteiro pode ser treinado usando apenas o algoritmo de backpropagation. Na prática, o treinamento é implementado de forma iterativa usando descida de gradiente estocástico em *minibatches*.

O procedimento itera entre k passos de otimização de D e um passo de otimização de G.

Passos de Otimização de D

Em k passos, o discriminador é atualizado para ascender seu gradiente estocástico (para maximizar $V(D,G)$).

A expressão para a atualização do gradiente estocástico do discriminador (∇_{θ_d}) baseada em um *minibatch* de m amostras de ruído $z^{(1)},\ldots,z^{(m)}$ e m exemplos reais $x^{(1)},\ldots,x^{(m)}$ é:

$$\nabla_{\theta_d}\frac{1}{m}\sum_{i=1}^{m}\left[\log D(x^{(i)})+\log\Big(1-D(G(z^{(i)}))\Big)\right].$$

Passo de Otimização de G

O gerador é atualizado uma vez para descender seu gradiente estocástico (para minimizar $V(D,G)$).

A expressão para a atualização do gradiente estocástico do gerador (∇_{θ_g}) é:

$$\nabla_{\theta_g}\frac{1}{m}\sum_{i=1}^{m}\log\Big(1-D(G(z^{(i)}))\Big).$$

Modificação Prática da Função Objetivo de G

Na prática, a função objetivo original de G (minimizar $\log(1-D(G(z)))$) pode não fornecer um gradiente suficiente para G no início do aprendizado. Isso ocorre porque, quando G é fraco, D rejeita as amostras falsas com alta confiança, fazendo com que $\log(1-D(G(z)))$ sature.

Para garantir gradientes mais fortes no início, G é, em vez disso, treinado para maximizar $\log D(G(z))$. Esta função objetivo alternativa resulta no mesmo ponto fixo da dinâmica entre G e D, mas fornece gradientes significativamente mais fortes quando o modelo está começando a aprender.

Resultados Teóricos

A análise teórica da convergência das GANs é realizada em um ambiente não-paramétrico, onde se assume que G e D têm capacidade infinita, estudando a convergência no espaço das funções de densidade de probabilidade.

Otimilidade Global de $p_g = p_{\text{data}}$

Proposição 1: O Discriminador Ótimo

Para um Gerador G fixo, a Proposição 1 define o Discriminador Ótimo, D^*_G.

O objetivo do discriminador é maximizar a função valor $V(G,D)$:

$$V(G,D)=\int_x p_{\text{data}}(x)\log(D(x))dx+\int_z p_z(z)\log(1-D(G(z)))dz$$

Reescrevendo a integral sobre z como uma integral sobre x usando $p_g(x)$, a expressão a ser maximizada é:

$$V(G,D) = \int_x [p_{\text{data}}(x)\log(D(x)) + p_g(x)\log(1-D(x))]\,dx \quad \text{(Equação 3)}$$

A função sob a integral atinge seu máximo no intervalo $$ em $\frac{a}{a+b}$ para a forma $a \log(y) + b \log(1-y)$.

Portanto, o discriminador ótimo $D^*_G(x)$ é:

$$D^*G(x) = \frac{p}{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)} \quad \text{(Equação 2)}$$

O critério de treinamento para D pode ser interpretado como a maximização da verossimilhança logarítmica para estimar a probabilidade condicional $P(Y = y|x)$, onde $Y = 1$ significa que x veio de p_{data}, e $Y = 0$ significa que x veio de p_g.

Teorema 1: O Ótimo Global do Jogo

Ao substituir o discriminador ótimo $D^*_G(x)$ na função valor $V(G,D)$, obtemos o **critério de treinamento virtual** $C(G)$:

$$C(G) = \max_D V(G,D) = E_{x \sim p_{\text{data}}}[\log D^G(x)] + E{x \sim p_g}[\log(1-D^_G(x))] \quad \text{(Equação 4)}$$

O Teorema 1 afirma que o mínimo global de $C(G)$ é alcançado se e somente se $p_g = p_{\text{data}}$.

Prova e Relação com JSD:

1. Se $p_g = p_{\text{data}}$, então $D^*G(x) = \frac{p}{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{data}}(x)} = 1/2$.
2. Neste ponto, $C(G)$ atinge o valor de $-\log 4$:

$$C(G) = E_{x\sim p_{\text{data}}}[\log 1/2] + E_{x\sim p_{\text{data}}}[\log 1/2] = \log(1/2) + \log(1/2) = -\log 4.$$

3. Para provar que este é o mínimo, $C(G)$ é reescrito usando a **Divergência de Jensen–Shannon (JSD)**:

$$C(G) = -\log(4) + \mathrm{KL}\left(p_{\text{data}} \parallel \frac{p_{\text{data}} + p_g}{2}\right) + \mathrm{KL}\left(p_g \parallel \frac{p_{\text{data}} + p_g}{2}\right) \quad \text{(Equação 5)}$$

Reconhecendo a JSD, a equação se simplifica para:

$$C(G) = -\log(4) + 2 \cdot JSD(p_{\text{data}} \parallel p_g) \quad \text{(Equação 6)}$$

Como a JSD é sempre não negativa ($JSD \geq 0$) e é zero somente quando $p_{\text{data}} = p_g$, fica provado que o mínimo global $C^* = -\log(4)$ é alcançado se e somente se o gerador replicar perfeitamente a distribuição dos dados.

Convergência do Algoritmo 1

Proposição 2: Convergência de p_g para p_{data}

A Proposição 2 aborda a convergência do Algoritmo 1 (o procedimento iterativo de treinamento).

A Proposição afirma que, se G e D tiverem capacidade suficiente e, em cada etapa do Algoritmo 1, o discriminador for permitido atingir seu ótimo ($D^*G)dadoG, ep_g foratualizadoparamelhorarocritérioC(G), entãop_gconvergeparap{\text{data}}$.

A base matemática é a convexidade:

O critério $U(p_g, D)$ (onde U é o mesmo que V) é considerado como uma função de p_g. A função $\sup_D U(p_g, D)$ (que é $C(G)$) é convexa em p_g e possui um ótimo global único, como provado no Teorema 1.

A atualização do gerador corresponde a um passo de descida de gradiente para p_g na solução ótima de D. Com atualizações suficientemente pequenas de p_g, o algoritmo de descida de gradiente converge para o ótimo global.

É importante notar que, na prática, em vez de otimizar p_g, otimizamos os parâmetros θ_g que definem $G(z; \theta_g)$, e o uso de MLPs introduz múltiplos pontos críticos no espaço de parâmetros.

Analogia para o Ótimo Global:

Pense no processo de treinamento GAN como um cabo de guerra (o jogo minimax) entre dois times G e D que estão puxando a distribuição gerada p_g em direção à distribuição real p_{data}. O ponto de equilíbrio matemático é alcançado quando a corda está perfeitamente centralizada: p_g é idêntica a p_{data}. Neste momento, o "juiz" D não consegue ver para que lado a corda está se movendo, pois a probabilidade de ambas as distribuições é a mesma, resultando em $D(x) = 1/2$ em todos os lugares. A relação com a Divergência de Jensen–Shannon (JSD) formaliza isso, pois a JSD mede a "distância" entre as duas distribuições, e o jogo garante que essa distância seja minimizada até zero.

Avaliação dos experimentos

Foram realizados testes realizados no artigo para demonstrar o potencial do *framework* GAN através da avaliação quantitativa e qualitativa das amostras geradas. Os modelos foram treinados em vários nos seguintes conjuntos de dados: MNIST, Toronto Face Database (TFD) e CIFAR-10.

Os modelos Generativos G e Discriminativos D foram definidos usando MLPs

- Rede Geradora (G): Utilizou uma mistura de ativações ReLU e ativações sigmoides. O ruído foi usado como entrada apenas para a camada mais baixa da rede geradora.
- Rede Discriminativa (D): Utilizou ativações maxout. O algoritmo Dropout foi aplicado durante o treinamento da rede discriminativa.
- Amostragem: As amostras geradas são sorteios aleatórios justos (não selecionados) e não correlacionados, pois o processo de amostragem não depende da mistura da cadeia de Markov.

Avaliação Quantitativa: Estimativa de Verossimilhança

Devido à natureza implícita do modelo generativo (p_g), a verossimilhança exata ($p(x)$) não é tratável. Para estimar a probabilidade do conjunto de dados de teste sob p_g, os autores utilizaram a técnica de janela de Parzen Gaussiana.

- Procedimento: Consiste em ajustar uma janela de Parzen Gaussiana às amostras geradas por G e então relatar a verossimilhança logarítmica sob esta distribuição.
- Parâmetro σ: O parâmetro σ das Gaussianas é obtido através de validação cruzada no conjunto de validação.
- Limitações: Este método de estimativa de verossimilhança possui variância razoavelmente alta e não funciona bem em espaços de alta dimensão, embora seja o melhor método disponível para modelos

que podem amostrar, mas não estimar a verossimilhança diretamente.
- Resultados: Os resultados (Tabela 1) comparam as GANs com outros modelos (DBN, Stacked CAE, Deep GSN), mostrando que as GANs são competitivas, especialmente na versão de valores reais do conjunto de dados MNIST.

Avaliação Qualitativa

As amostras visuais geradas são consideradas competitivas com os melhores modelos generativos na literatura, destacando o potencial do *framework* adversarial. As visualizações também demonstram que o modelo não memorizou o conjunto de treinamento, comparando as amostras geradas com o exemplo de treinamento mais próximo.

Vantagens

As vantagens são majoritariamente computacionais e simplificam o processo de treinamento e amostragem em comparação com modelos como Máquinas de Boltzmann (RBMs/DBMs) ou Redes Generativas Estocásticas (GSNs):

1. Sem Cadeias de Markov: O *framework* nunca requer cadeias de Markov. O processo de amostragem em GANs não depende da mistura da cadeia de Markov.
2. Uso Exclusivo de Backpropagation: Apenas o algoritmo de backpropagation é necessário para obter os gradientes e treinar o sistema inteiro (assumindo MLPs).
3. Inexistência de Inferência Durante o Aprendizado: Nenhuma inferência é necessária durante o aprendizado.
4. Flexibilidade de Design: Uma ampla variedade de funções diferenciáveis pode ser incorporada ao modelo.

Além das vantagens computacionais, os modelos adversariais possuem benefícios inerentes à sua capacidade de representação:

1. Representação de Distribuições Degeneradas: As redes adversariais têm a capacidade de representar distribuições muito nítidas, até mesmo degeneradas. Isso contrasta com métodos baseados em cadeias de Markov, que exigem que a distribuição seja de alguma forma "embaçada" (*blurry*) para que as cadeias consigam misturar-se entre os modos.
2. Benefício Estatístico Indireto: Pode haver uma vantagem estatística porque a rede geradora não é atualizada diretamente com exemplos de dados, mas apenas com gradientes que fluem através do discriminador. Isso garante que os componentes da entrada não sejam copiados diretamente para os parâmetros do gerador.

A Tabela 2 nas fontes resume a comparação do *framework* de redes generativas adversariais com outras abordagens de modelagem profunda, destacando os desafios enfrentados por cada técnica (como a necessidade de MCMC para amostragem ou dificuldade na avaliação de $p(x)$).

Desvantagens

As desvantagens estão associadas principalmente à forma como a distribuição é representada e como as redes devem ser treinadas em conjunto:

1. Ausência de Representação Explícita de $p_g(x)$: Não há uma representação explícita da distribuição de probabilidade gerada, $p_g(x)$. A probabilidade de $p(x)$ só pode ser aproximada, por exemplo, através

da estimativa de densidade de Parzen.
2. Sincronização de D e G: O discriminador D deve ser bem sincronizado com o gerador G durante o treinamento.
3. Risco de Colapso do Gerador: Deve-se evitar que G seja treinado "demais" sem atualizar D. Caso contrário, o modelo pode incorrer no "cenário Helvetica", onde G colapsa muitos valores de z para o mesmo valor de x, resultando em amostras geradas com pouca diversidade para modelar p_{data}.

Modelos Derivados e Aprimoramentos

Com o crescimento de sua popularidade, há uma infinidade de variações de GAN. Como são inúmeras, iremos falar sobre:

- DCGANS
- WGANS
- cGAN
- Pix2Pix
- CycleGAN
- SR-GANS
- Pro-GAN
- Style-GAN

DCGAN (Deep Convolutional GAN)

A DCGAN (Deep Convolutional GAN) é um tipo específico de GAN, proposta por *Radford et al.* (2015), projetada para resolver limitações das GANs originais. Enquanto as primeiras GANs usavam principalmente MLPs totalmente conectadas, a DCGAN introduziu o uso sistemático de redes neurais convolucionais profundas (CNNs), tornando-se especialmente adequada para geração de imagens.

A adoção de CNNs não apenas melhorou a qualidade visual das amostras geradas, mas também contribuiu significativamente para a estabilidade do treinamento, sendo considerada um dos modelos mais robustos entre os GANs clássicos.

Principais características da DCGAN

Gerador:

- Usa ConvTranspose2D (convoluções transpostas) para upsampling.
- Evita camadas totalmente conectadas depois da primeira camada.
- Batch Normalization em quase todas as camadas (exceto na última).
- Ativação ReLU em todas as camadas internas.
- Saída com tanh, para normalizar imagens entre [-1, 1].

Discriminador:

- Convoluções 2D com stride para downsampling — sem pooling.
- LeakyReLU como função de ativação principal.
- Batch Normalization (com exceção da primeira camada).
- Camada final com sigmoid (para probabilidade de “real” vs “fake”).

Função de perda:

DCGAN tradicional usa Binary Cross-Entropy Loss (BCE): $\mathcal{L}_D = -\mathbb{E}[\log(D(x))] - \mathbb{E}[\log(1 - D(G(z)))]$
$\mathcal{L}_G = -\mathbb{E}[\log(D(G(z)))]$

Pela implementação original, trata-se de BCE com logits, na prática implementada como `BCELoss`.

Por que a DCGAN é melhor que a GAN original?Fewer Parameters, Better Generalization

A GAN original:

- Usava MLPs, que ignoram a estrutura espacial das imagens.
- Era extremamente instável.
- Gerava imagens borradas, sem consistência global.
- Era mais propensa a *mode collapse*.

A DCGAN introduziu boas práticas estruturais que praticamente redefiniram o padrão para modelos geradores baseados em CNN:

Exploração da estrutura espacial

As convoluções capturam padrões locais (bordas, texturas), permitindo ao gerador produzir imagens muito mais realistas do que MLPs.

Estabilidade de treinamento muito maior

O uso combinado de:

- BatchNorm
- ReLU / LeakyReLU
- Conv / ConvTranspose

Reduz drasticamente problemas como *vanishing gradients* e facilita o aprendizado.

Representações latentes mais ricas

A DCGAN descobriu que a *latent space* se torna **contínua e semântica**, permitindo:

- interpolação suave,
- manipulação de atributos,
- controle de estilo (insight fundamental para StyleGAN anos depois).

Menos parâmetros, melhor generalização

CNNs têm:

- peso compartilhado,
- filtro espacial,
- menos redundância. Isso melhora a generalização e a qualidade dos samples.

Arquitetura padronizada

Um dos maiores méritos: pela primeira vez havia uma “receita” estável para treinar GANs. Quase todos os trabalhos subsequentes (WGAN, CGAN, StyleGAN) se basearam nos princípios da DCGAN.

WGAN (Wasserstein Generative Adversarial Network)

O WGAN, introduzido no paper "Wasserstein GAN" (Arjovsky et al., 2017), representa um avanço significativo no treinamento de Redes Adversariais Generativas (GANs), principalmente ao substituir a divergência de Jensen-Shannon (usada na GAN original) pela distância de Wasserstein (também conhecida como Earth Mover's Distance - EMD) para medir a diferença entre a distribuição de dados reais (P_r) e a distribuição gerada (P_g).

A distância de Wasserstein mede o esforço mínimo necessário (em termos de "custo" de transporte) para transformar a distribuição P_g na distribuição P_r. Formalmente, a distância de Wasserstein-1 entre P_r e P_g é definida como:

$$W(P_r, P_g) = \inf_{\gamma \in \Pi(P_r, P_g)} \mathbb{E}_{(x,y)\sim\gamma}[||x - y||]$$

onde $\Pi(P_r, P_g)$ é o conjunto de todas as distribuições de probabilidade conjuntas $\gamma(x, y)$ cujas marginais são P_r e P_g, respectivamente.

Melhorias do WGAN

O uso da distância de Wasserstein resultou em diversas melhorias cruciais para a estabilidade e qualidade do treinamento das GANs:

- Melhor Estabilidade no Treinamento: A EMD é contínua e diferenciável em quase todos os pontos, mesmo quando as distribuições não se sobrepõem (o que é comum no início do treinamento de GANs), fornecendo gradientes mais significativos para o gerador.
- Solução para Vanishing Gradients: Na GAN original, quando o discriminador se torna "muito bom", a função de perda de *cross-entropy* satura e os gradientes se aproximam de zero (problema de *vanishing gradients*), impedindo o aprendizado do gerador. A WGAN mitiga isso.
- Mitigação de *Mode Collapse*: A função de perda baseada na EMD penaliza o modelo de forma mais suave, permitindo que o gerador explore melhor o espaço de dados, reduzindo a tendência de produzir apenas um subconjunto limitado de amostras (fenômeno de *mode collapse*).

Arquitetura e Função de Perda (Loss)

O Crítico (Critic)

O Discriminador** é renomeado para Crítico no WGAN. Em vez de produzir uma probabilidade binária (real ou falsa, 0 ou 1), o Crítico (C) é treinado para retornar uma pontuação que aproxima a distância de Wasserstein. O objetivo é treinar o Crítico para aproximar o seguinte valor:

$$W(P_r, P_g) = \sup_{C \in \mathcal{L}*1} *mathbb{E}*{x \sim P_r} [C(x)] - \mathbb{E}_{z \sim P(z)} [C(G(z))]$$

onde $\mathcal{L}_1$ é o conjunto de funções 1-Lipschitz.

Restrição de Lipschitz

Para que o Crítico aproxime corretamente a EMD, ele deve satisfazer a restrição de Lipschitz com constante 1 ($||C(x_1) - C(x_2)|| \leq ||x_1 - x_2||$).

No WGAN original, essa restrição é imposta por um método simples, mas problemático: o corte de peso (weight clipping). Isso consiste em forçar os pesos da rede do Crítico a permanecerem dentro de um intervalo fixo, como $[-c, c]$, após cada atualização.

Funções de Perda do WGAN

A função de perda para o Crítico (L_C): $$L_C = \mathbb{E}*{z \sim P(z)} [C(G(z))] - \mathbb{E}*{x \sim P_r} [C(x)]$$

O Crítico é treinado para maximizar essa perda (aproximando $W(P_r, P_g)$).

A função de perda para o Gerador (L_G):

$$L_G = -\mathbb{E}_{z \sim P(z)} [C(G(z))]$$

O Gerador é treinado para minimizar essa perda.

WGAN-GP (Wasserstein GAN with Gradient Penalty)

A técnica de corte de peso do WGAN original foi identificada como um problema, pois tende a forçar a distribuição de pesos a se concentrar em valores extremos e pode limitar a capacidade de modelagem do Crítico.

O WGAN-GP (Gulrajani et al., 2017) resolve isso introduzindo uma penalidade de gradiente (Gradient Penalty - GP) à função de perda do Crítico, que impõe a restrição de 1-Lipschitz de forma muito mais eficiente e estável.

Função de Perda do Crítico no WGAN-GP

A perda do Crítico é modificada com a adição do termo de penalidade: $$ L_{C_{GP}} = \underbrace{\mathbb{E}*{z \sim P(z)} [C(G(z))] - \mathbb{E}*{x \sim P_r} [C(x)]}*{\text{Perda de Wasserstein}} + \underbrace{\lambda \mathbb{E}*{\hat{x} \sim P_{\hat{x}}} [(||\nabla_{\hat{x}} C(\hat{x})||_2 - 1)^2]}*{\text{Penalidade de Gradiente}} $$

- λ: É um hiperparâmetro de penalidade (geralmente $\lambda = 10$).
- $\hat{x} \sim P_{\hat{x}}$: $\hat{x}$ são amostras obtidas pela interpolação linear entre um ponto da distribuição real (x) e um ponto da distribuição gerada ($G(z)$): $\hat{x} = \epsilon x + (1 - \epsilon) G(z)$, onde $\epsilon \sim U(0, 1)$.
- $\nabla_{\hat{x}} C(\hat{x})$: É o gradiente do Crítico em relação à entrada $\hat{x}$.

O termo de penalidade força a norma do gradiente do Crítico a ser próxima de 1 ao longo das amostras interpoladas, garantindo que a restrição de 1-Lipschitz seja satisfeita de forma robusta e sem limitar a capacidade do modelo.

O WGAN-GP é, em geral, o método de treinamento preferido para GANs, pois oferece treinamento estável e alta qualidade de amostras geradas sem a necessidade de weight clipping e seus problemas associados.

cGAN (Conditional Generative Adversarial Network)

O cGAN (Conditional Generative Adversarial Network) é uma extensão da arquitetura GAN original, introduzida por Mirza e Osindero em 2014. A principal inovação é a inclusão de informação condicional (condition, y) tanto no Gerador (G) quanto no Discriminador (D), permitindo que o modelo gere dados de saída específicos com base em uma *entrada de condição*.

Sem essa condição, uma GAN tradicional gera amostras aleatórias que refletem a distribuição geral do conjunto de dados de treinamento, mas não permite o controle sobre as características do dado gerado (por exemplo, qual dígito um MNIST GAN deve gerar). Com o cGAN, essa capacidade de controle é adicionada.

Arquitetura e Fluxo

No cGAN, a informação condicional (y) pode ser de qualquer tipo: uma *label* de classe, dados de outra modalidade (como um mapa semântico), ou até mesmo parte dos dados de entrada.

O Gerador

O Gerador não recebe apenas o vetor de ruído (z) como entrada, mas também a informação condicional (y). Ele aprende o mapeamento da seguinte forma:

$$G : z, y \rightarrow x'$$

Onde x' é a amostra gerada. Tipicamente, o ruído z e a condição y são concatenados e alimentados na rede neural.

O Discriminador

O Discriminador recebe tanto a amostra de dados (x ou x') quanto a informação condicional (y) correspondente. Ele é treinado para determinar se a amostra (x) é real ou falsa dado o contexto y.

$$D : x \text{ ou } x', y \rightarrow \text{probabilidade } (\text{real vs. falsa})$$

A condição y é concatenada com o dado de entrada (x ou x') nas camadas de entrada ou em camadas intermediárias do Discriminador.

Função de Perda

A função de perda do cGAN é uma modificação direta da função de perda da GAN original (baseada em *cross-entropy*), onde a dependência da condição y é explicitamente adicionada.

Objetivo Mínimo-Máximo do cGAN

O cGAN busca resolver o seguinte jogo de dois jogadores:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim P_{data}(x)} [\log D(x | y)] + \mathbb{E}_{z \sim P_z(z)} [\log (1 - D(G(z | y)))]$$

Onde:

- $D(x|y)$: É a probabilidade de que x seja real, dado a condição y.
- $G(z|y)$: É a amostra gerada pelo Gerador, dada a condição y e o ruído z.

Detalhamento da Perda

1. Perda do Discriminador (L_D) 🤫* O Discriminador tenta maximizar essa função, ou seja, atribuir alta probabilidade aos dados reais condicionados (x, y) e baixa probabilidade aos dados falsos condicionados ($G(z, y), y$). $$\max_D V(D, G) = \mathbb{E}{x \sim P{data}(x)} [\log D(x, y)] + \mathbb{E}_{z \sim P_z(z)} [\log (1 - D(G(z, y), y))]$$

2. Perda do Gerador (L_G) 🤫* O Gerador tenta minimizar essa função, ou seja, fazer com que D atribua alta probabilidade às amostras falsas ($G(z, y)$) para a condição y correspondente.

$$\min_G V(D, G) = \mathbb{E}_{z \sim P_z(z)}[\log(1 - D(G(z, y), y))]$$

Aplicações Chave

O cGAN é a base para muitas aplicações de visão computacional e processamento de linguagem que requerem geração controlada, onde a saída não é apenas aleatória, mas deve corresponder a uma entrada ou um desejo específico do usuário.

Aplicação	Condição (y)	Saída Gerada (x')
Geração de Imagens	Rótulo da Classe ("Gato", "Carro")	Imagem correspondente ao Rótulo
Tradução Imagem-a-Imagem (Pix2Pix)	Mapa Semântico (ex: contornos de rua)	Imagem realística correspondente (ex: foto da rua)
Geração de Texto	Primeira parte de uma frase	Continuação da frase

Pix2Pix (Conditional Adversarial Networks for Image-to-Image Translation)

O Pix2Pix é um framework de tradução de imagem-a-imagem proposto por Isola et al. (2017), construído diretamente sobre a arquitetura cGAN (Conditional GAN). O seu principal objetivo é aprender um mapeamento da imagem de entrada (x) para a imagem de saída correspondente (y) com base em uma condição.

Requisito Fundamental: Dados Pareados (Paired Data)

O Pix2Pix exige dados de treinamento pareados, onde cada imagem de entrada (Domínio A) deve ter um correspondente exato e alinhado no Domínio B.

Domínio A (Entrada x)	Domínio B (Saída y)	Exemplos de Aplicação
Mapa de Contorno/Esboço	Foto Real	Geração de imagem a partir de desenho
Mapa Semântico	Foto de Rua	Síntese de paisagem urbana
Imagem Infravermelha	Imagem RGB	Conversão de modalidade

Arquitetura e Componentes Chave

1. Gerador (G - U-Net):

- Em vez de uma rede totalmente convolucional padrão, o Pix2Pix usa uma arquitetura U-Net como Gerador.
- O U-Net é uma rede codificador-decodificador com conexões de salto (skip connections) que ligam diretamente as camadas do codificador às camadas correspondentes do decodificador.
- Vantagem: Essas conexões permitem que informações de baixa frequência (como contornos e estrutura global) ignorem a garganta da rede e sejam transferidas diretamente para a saída, ajudando a manter a estrutura da imagem de entrada e evitando o "borrão" (blurring) na imagem gerada.

2. Discriminador (D - PatchGAN):

- O Discriminador utiliza uma arquitetura chamada PatchGAN.
- Em vez de classificar a imagem inteira como real ou falsa (o que é trivial em muitos casos de tradução), o PatchGAN classifica patches (pedaços) de $N \times N$ da imagem de saída como reais ou falsos.
- Vantagem: Isso força o Gerador a focar em detalhes de alta frequência (textura e estilo) em regiões localizadas da imagem, resultando em saídas mais nítidas e realistas.

Função de Perda Total

A perda do Pix2Pix é uma combinação da perda adversarial do cGAN e uma perda de reconstrução L1 (ou L2):

$$L_{Total}(G, D) = L_{cGAN}(G, D) + \lambda L_{L1}(G)$$

1. Perda Adversarial (L_{cGAN}): A perda padrão do cGAN, que incentiva o Gerador a produzir amostras que pareçam reais para o Discriminador, *condicionadas à entrada* x. $$L_{cGAN}(G, D) = \mathbb{E}*{x, y} [\log D(x, y)] + \mathbb{E}*{x, z} [\log (1 - D(x, G(x, z)))]$$
2. Perda L1 (L_{L1} - Reconstrução): Mede o erro absoluto médio entre a imagem alvo real (y) e a imagem gerada ($G(x)$). Esta perda garante que a imagem gerada não seja apenas realista, mas também corresponda à imagem alvo.

$$L_{L1}(G) = \mathbb{E}_{x,y}[||y - G(x)||_1]$$

- O hiperparâmetro λ (comumente $\lambda = 100$) controla o peso da perda L1, que é essencial para manter a fidelidade da imagem traduzida.

CycleGAN (Cycle-Consistent Adversarial Networks)

O CycleGAN, proposto por Zhu et al. (2017), é uma solução elegante para o problema da tradução de imagem-a-imagem quando **dados pareados são escassos ou impossíveis de obter**.

Inovação Fundamental: Dados Não Pareados (Unpaired Data)

O CycleGAN pode aprender a tradução entre duas coleções de imagens, X e Y (Domínio A e Domínio B), sem que haja uma correspondência 1:1 entre as imagens.

Domínio X	Domínio Y	Exemplos de Aplicação
Cavalos	Zebras	Conversão de espécies

Domínio X	Domínio Y	Exemplos de Aplicação
Fotos (Verão)	Fotos (Inverno)	Transferência de estação
Fotos (Estilo A)	Pinturas (Estilo B)	Transferência de estilo artístico

Arquitetura

O CycleGAN opera com duas GANs completas que se complementam, formando um ciclo:

1. GAN $X \to Y$:

- Gerador G: Mapeia imagens de X para Y. ($G : X \to Y$)
- Discriminador D_Y: Distingue entre imagens reais em Y e imagens geradas por G.

2. GAN $Y \to X$:

- Gerador F: Mapeia imagens de Y para X. ($F : Y \to X$)
- Discriminador D_X: Distingue entre imagens reais em X e imagens geradas por F.

Função de Perda Total e Consistência Cíclica

A função de perda do CycleGAN é composta por três termos:

$$L_{Total} = L_{GAN}(G, D_Y) + L_{GAN}(F, D_X) + \lambda L_{cyc}(G, F)$$

1. Perda Adversarial (L_{GAN}): É aplicada a ambas as GANs para garantir que as imagens geradas pelos Geradores G e F sejam indistinguíveis de imagens reais em seus domínios alvo.

2. Perda de Consistência Cíclica (L_{cyc}): Este é o componente mais crucial. Ele impõe que a tradução e a re-tradução de uma imagem devem resultar em algo próximo à imagem original. Isso evita que os Geradores "trapaceiem" mapeando todas as imagens de um domínio para um único ponto no outro (evitando *mode collapse*).

    - Ciclo Direto ($X \to Y \to X$): Se traduzirmos x para Y usando G, e depois traduzirmos de volta para X usando F, o resultado deve ser x.

    $$L_{cyc_forward} = \mathbb{E}_x[||F(G(x)) - x||_1]$$

    - Ciclo Reverso ($Y \to X \to Y$): Se traduzirmos y para X usando F, e depois traduzirmos de volta para Y usando G, o resultado deve ser y.

    $$L_{cyc_backward} = \mathbb{E}_y[||G(F(y)) - y||_1]$$

    A Perda de Consistência Cíclica é a soma das duas: $L_{cyc}(G, F) = L_{cyc_forward} + L_{cyc_backward}$.

3. Perda de Identidade (*Identity Loss* - opcional): Ajuda a preservar a cor e a composição se a imagem de entrada já pertencer ao domínio alvo. $L_{id}(G, F) = \mathbb{E}_{y} [||G(y) - y||_1] + \mathbb{E}_{x} [||F(x) - x||_1]$.

O CycleGAN permitiu a aplicação de tradução de imagem-a-imagem em inúmeros domínios onde a obtenção de dados pareados seria proibitiva.

Os modelos ProGAN e StyleGAN, ambos desenvolvidos pela NVIDIA (por Tero Karras et al.), marcaram saltos de qualidade e resolução sem precedentes na geração de imagens fotorrealistas por GANs. O StyleGAN, na verdade, é uma evolução direta do ProGAN, incorporando sua metodologia de treinamento e adicionando uma nova arquitetura de gerador baseada em estilos.

StyleGAN (Style-Based Generator Architecture)

O StyleGAN (Karras et al., 2018) é construído sobre a base do ProGAN, mas introduz uma arquitetura de Gerador totalmente nova inspirada nas técnicas de Transferência de Estilo. O objetivo é alcançar o desemaranhamento dos fatores latentes, permitindo um controle intuitivo sobre os atributos da imagem gerada.

Inovações Arquiteturais Chave

O StyleGAN modifica profundamente o Gerador tradicional (que recebe o vetor latente z na entrada) com três componentes principais:

Rede de Mapeamento

- Em vez de usar o vetor latente z (amostrado de uma Gaussiana) diretamente, o StyleGAN usa uma rede de mapeamento (uma MLP com 8 camadas) para transformar z em um vetor intermediário (w).
- O vetor w reside em um espaço latente intermediário ($\mathcal{W}$). Este espaço $\mathcal{W}$ é projetado para ser menos emaranhado (less entangled) do que o espaço Z, facilitando a separação dos fatores de variação (ex: pose, cor, identidade).

Injeção de Estilo

- O Gerador StyleGAN é renomeado para Rede de Síntese. Ele recebe um tensor constante como entrada (e não o vetor z ou w).
- O Gerador aplica os "estilos" (vetor w) em cada bloco de resolução usando a Normalização de Instância Adaptativa (AdaIN):

$$\text{AdaIN}(x_i, w) = y_{s,i} \frac{x_i - \mu(x_i)}{\sigma(x_i)} + y_{b,i}$$

  onde x_i é o mapa de *features* de entrada, e $y_{s,i}$ (escala) e $y_{b,i}$ (viés) são obtidos a partir do vetor de estilo w.
- Vantagem: A injeção de estilo em diferentes resoluções permite controlar diferentes níveis de detalhe:
  - Estilos de Resolução Baixa: Controlam atributos de alto nível e globais (ex: pose, formato do rosto, tipo de cabelo).
  - Estilos de Resolução Alta: Controlam detalhes finos e estocásticos (ex: cor do cabelo, sardas, rugas).

Injeção de Ruído (*Noise Injection*)

- Uma fonte de ruído estocástico e não correlacionado é adicionada a cada bloco de síntese, separadamente.
- Vantagem: Isso é usado para controlar a variação estocástica na imagem (ex: colocação exata de mechas de cabelo, texturas finas) sem afetar os atributos de alto nível controlados pelo estilo (w).

StyleGAN2 (2020)

O StyleGAN1, apesar de produzir resultados incríveis, possuía artefatos visuais perceptíveis, principalmente gotas de água (droplet artifacts) e texturas estáticas. O StyleGAN2 foi desenvolvido principalmente para eliminar esses artefatos e melhorar a qualidade geral da imagem.

Principais Mudanças e Inovações

- Remoção de Artefatos de Estereótipos (Droplets): Os artefatos eram causados em parte pela Normalização de Instância Adaptativa (AdaIN) e pela normalização de caminho de sinal dentro do Gerador.
    - Substituição do AdaIN: O StyleGAN2 removeu a normalização de média/variância do AdaIN. A operação foi simplificada, onde apenas a modulação de estilo e a desmodulação são usadas.
    - Normalização de Caminho (Path Length Regularization): Introduziu uma nova penalidade na função de perda para encorajar um mapeamento mais suave e melhor *desemaranhamento* no espaço latente. Isso significa que uma pequena alteração no vetor W resulta em uma alteração proporcional e bem-comportada na imagem, eliminando as texturas estáticas.
- Melhoria na Qualidade e Coerência: A remoção dos artefatos e a nova regularização resultaram em imagens mais nítidas e com melhor coerência em larga escala.
- Novo Design da Arquitetura: O Gerador foi redesenhado para ser mais estável, eliminando as conexões de desvio (skip connections) do ProGAN, que eram usadas no StyleGAN1.

Considerações Finais

- As GANs estabeleceram um novo e poderoso paradigma para a modelagem generativa, enquadrando-a como um jogo minimax de soma zero entre duas redes
- Essa abordagem evita os cálculos de probabilidade complexos e frequentemente intratáveis exigidos por métodos anteriores
- As pesquisas subsequentes mais significativas se concentraram no controle e na aplicação
- Arquiteturas avançadas como o StyleGAN introduziram espaços latentes controláveis e desvinculados, transformando as GANs de simples geradoras de dados em usuários que permitem o controle semântico sobre o processo de geração