Projeto - Face Generation

Disciplina de Tópicos Avançados - Visão Computacional - SCC0910

Aluno	número USP
Bernardo Marques Costa	11795551
Rodrigo Lopes Assaf	11795530
Victor Henrique de Sa Silva	11795759

Descrição do projeto

Nesse projeto de Visão Computacional, optamos por apresentar a seguinte tarefa: geração de imagens de faces. A geração de imagens é uma das aplicações mais interessantes da visão computacional, pois permite criar novas imagens que podem ser usadas em diversas aplicações, como por exemplo arte, design, publicidade e até no treinamento de outros modelos.

Teremos duas etapas de análises: na primeira, atacaremos o problema utilizando redes neurais generativas adversariais (mais especificamente, Deep Convolutional GANs - DCGANs) e na segunda, tentaremos aplicar modelos de difusão para melhorar a geração dessas faces.

Com isso, nesse nosso trabalho, estudaremos sobre a criação de imagens de faces falsas, de pessoas que não existem, tendo como base um dataset composto por imagens reais de pessoas famosas.

Metodologia da Primeira Iteração

Para este trabalho, utilizamos a base de dados CelebA (Liu et al., 2015), que contém mais de 200.000 imagens de celebridades.

Implementamos uma GAN baseada na arquitetura DCGAN, que consiste em uma rede geradora com cinco camadas convolucionais e uma rede discriminadora também com cinco camadas convolucionais. As imagens de entrada foram redimensionadas para o tamanho de 64x64 pixels e normalizadas entre -1 e 1. Usamos a função de ativação ReLU nas camadas convolucionais da rede geradora e a função de ativação LeakyReLU nas camadas convolucionais da rede discriminadora.

Treinamos a rede por 50 épocas com um *batch size* de 128 imagens e uma taxa de aprendizado de 0,0001. Para maior estabilidade, utilizamos o otimizador Adam com $\beta_1 = 0.5$ e $\beta_2 = 0.999$. Devido a característica de classificação, utilizamos como função de custo uma entropia cruzada.

Arquitetura Geradora

```
Generator(
  (main): Sequential(
    (0): ConvTranspose2d(100, 512, kernel_size=(4, 4), stride=(1, 1), bias=False)
    (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): ConvTranspose2d(512, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (4): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ReLU(inplace=True)
    (6): ConvTranspose2d(256, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (7): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (8): ReLU(inplace=True)
    (9): ConvTranspose2d(128, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (10): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (11): ReLU(inplace=True)
    (12): ConvTranspose2d(64, 3, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (13): Tanh()
  )
)
```

Arquitetura Discriminadora

```
Discriminator(
  (main): Sequential(
    (0): Conv2d(3, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
```

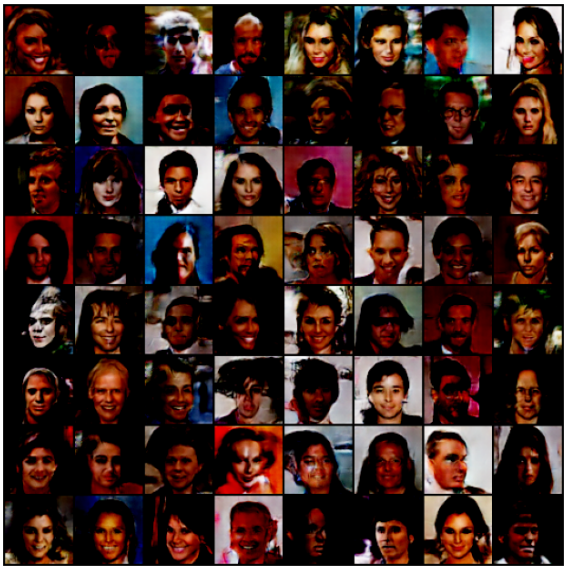
```
    (1): LeakyReLU(negative_slope=0.2, inplace=True)
    (2): Conv2d(64, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (3): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (4): LeakyReLU(negative_slope=0.2, inplace=True)
    (5): Conv2d(128, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (6): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (7): LeakyReLU(negative_slope=0.2, inplace=True)
    (8): Conv2d(256, 512, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (9): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (10): LeakyReLU(negative_slope=0.2, inplace=True)
    (11): Conv2d(512, 1, kernel_size=(4, 4), stride=(1, 1), bias=False)
    (12): Sigmoid()
  )
)
```

Resultados

Os resultados obtidos com a DCGAN proposta foram promissoras.

Final da primeira época

Final da décima época



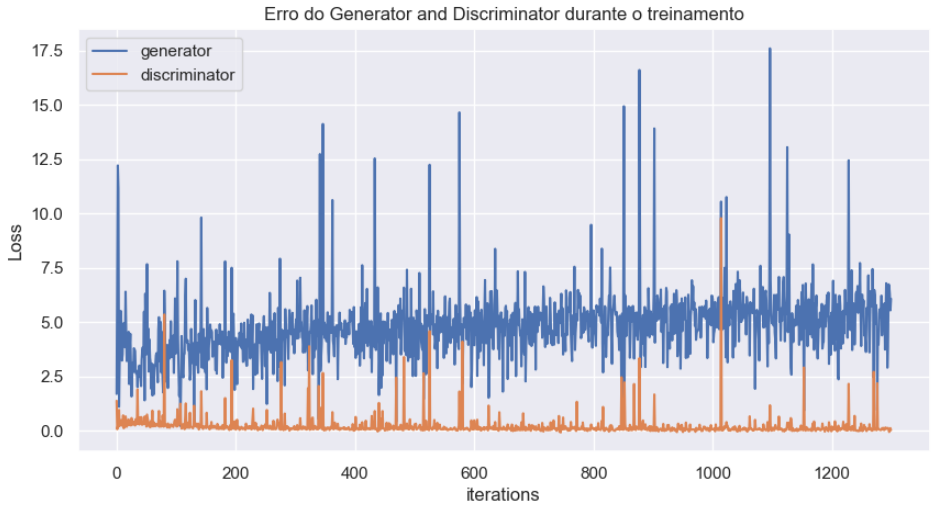
Final da vigésima época

Final da trigésima época

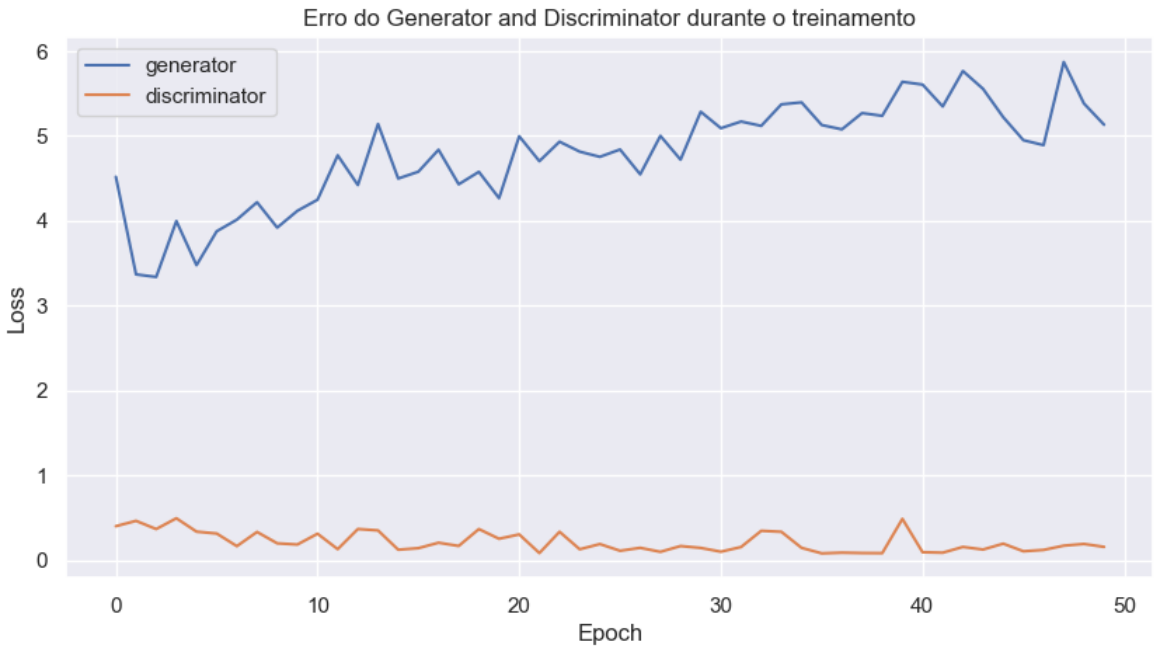
Final da quadragésima época

Final da quinquagésima época

As figuras acima mostram alguns exemplos de imagens geradas pela rede. Podemos ver que as imagens geradas são relativamente realistas com o passar das épocas e dos treinamentos e apresentam características de pessoas reais. Além disso, observamos que após um certo período, temos uma estagnação, e rede não evolui tanto referente à qualidade das imagens.



Avaliação dos erros ao passar das iterações



Erro médio dos modelos por época

Como podemos observar, a função de erro durante o treino tem um comportamento a princípio estranho, mas segundo implementações consideradas de GAN, é algo esperado. Além disso, os modelos conseguiram desempenhar o papel adversarial e criar um modelo gerador relativamente realista.

Trabalhos Futuros

Planejamos para a próxima iteração aplicar modelos de difusão para observar seu desempenho na criação de imagens. Nesse contexto, iremos observar o resultado de um modelo discretor entre os outputs da GAN da implementação atual e do modelo de difusão.

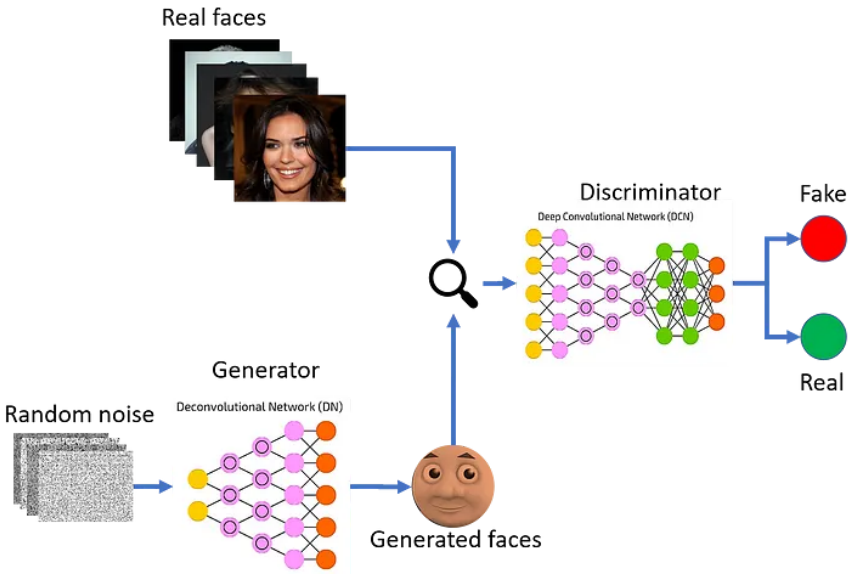
Repositório do projeto

O nosso projeto está disponível no Github.

Bibliografia

GANs

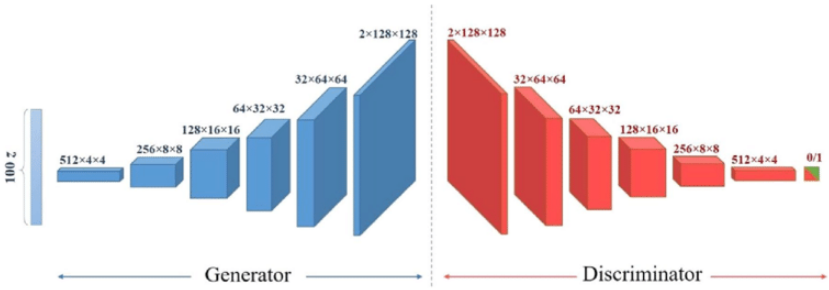
As GANs foram introduzidas por Goodfellow et al. (2014) e consistem em duas redes neurais, uma geradora e outra discriminadora, que são treinadas simultaneamente. A rede geradora gera imagens falsas, enquanto a rede discriminadora é treinada para distinguir imagens reais das imagens geradas. O treinamento é realizado em um processo iterativo em que a rede geradora busca gerar imagens cada vez mais realistas, enquanto a rede discriminadora busca distinguir entre imagens reais e falsas com maior precisão.



Existem várias arquiteturas de GANs que foram propostas na literatura, incluindo a DCGAN (Radford et al., 2016), a WGAN (Arjovsky et al., 2017) e a StyleGAN (Karras et al., 2019).

DCGANs

Na implementação que faremos aqui, utilizaremos de uma variação de rede neural adversarial proposta por Goodfellow. Escolhemos redes neurais convolucionais para implementar as redes geradora e discriminadora, técnica que foi proposta em DCGAN (Radford et al., 2016)



CelebFaces Attributes Dataset disponível no Kaggle

Para este trabalho, utilizamos a base de dados CelebA (Liu et al., 2015), que contém mais de 200.000 imagens de celebridades e disponibilizado na plataforma Kaggle