Aplicação do algoritmo DCGAN do pytorch para geração de imagens

Aluno: Eduardo Montovanelli Dalmaso
CPF:147.089.327-40

Resumo:

O objetivo do projeto era utilizar Deep Convolutional Generative Adversarial Networks (DCGANs) implementadas no PyTorch para gerar novas imagens de carros, baseadas em um conjunto de dados real de carros (The Car Connection Picture Dataset). A ideia era criar um modelo capaz de aprender as características visuais dos carros presentes no dataset e comparar com as imagens reais.

Pude observar que o modelo teve muita distorção ao representar imagens do interior dos carros, ou seja, de absorver e entender as essas características e boa apresentação de carros com fotos de vista lateral.

Métodos:

Utilizei o DCGAN do pytorch para explorar a geração de imagens de alta qualidade a partir de um dataset específico, meu objetivo principal era avaliar a capacidade do DCGAN em gerar imagens que fossem o mais próximo possível das imagens reais do meu dataset escolhido, ou seja, alcançar uma alta fidelidade visual.

O experimento foi utilizar uma base de exemplo do uso do DCGAN do pytorch e adaptá-lo ao uso do meu dataset. Ajustando os hiperparâmetros e épocas conforme testes. O dataset em questão é o **the-car-connection-picture-dataset** obtido no kaggle, contempla 60 mil imagens de diferentes veículos externo e interiores.

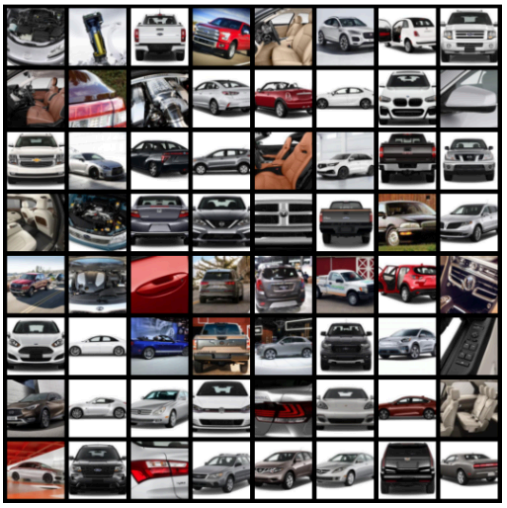

Fig 1- Dataset

Nesse dataset, eu redimensionei as imagens para 64x64 pixels e normalizei utilizando da biblioteca do pytorch, ImageFolder como mostrado abaixo:

```
# Carregar as imagens usando ImageFolder
dataset = dset.ImageFolder(root=dataroot,
                           transform=transforms.Compose([
                               transforms.Resize(image_size),
                               transforms.CenterCrop(image_size),
                               transforms.ToTensor(),
                               transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5)),
                           ]))
```

Fig 2 - Preparação dos dados

DCGAN:

Primeiro vamos entender o GAN (Redes Gerativas Adversárias) são um framework para ensinar um modelo de deep learning a capturar a distribuição dos dados de treinamento, para que possamos gerar novos dados a partir dessa mesma distribuição. As GANs foram inventadas por Ian Goodfellow em 2014 e descritas pela primeira vez no artigo **"Generative Adversarial Nets"**. Elas são compostas por dois modelos distintos: um **gerador** e um **discriminador**. O trabalho do gerador é criar imagens 'falsas' que se pareçam com as imagens de treinamento. O trabalho do discriminador é analisar uma imagem e dizer se ela é uma imagem real do treinamento ou uma imagem falsa criada pelo gerador. Durante o treinamento, o gerador está constantemente tentando enganar o discriminador, gerando imagens falsas cada vez melhores, enquanto o discriminador está se aprimorando para se tornar um melhor "detetive" e classificar corretamente as imagens reais e falsas. O equilíbrio desse "jogo" acontece quando o gerador consegue criar falsificações perfeitas, que parecem ter vindo diretamente dos dados de treinamento, e o discriminador fica incerto, com 50% de confiança, se a imagem gerada é real ou falsa.

D(G(z)) é a probabilidade (escalar) de que a saída do gerador G seja uma imagem real. Como descrito no artigo de Goodfellow, D e G jogam um jogo minimax em que D tenta maximizar a probabilidade de classificar corretamente as imagens reais e falsas (logD(x)\log D(x)logD(x)), enquanto GGG tenta minimizar a probabilidade de que D preveja que suas saídas são falsas (log(1−D(G(z)))\log(1 - D(G(z)))log(1−D(G(z)))). De acordo com o artigo, a função de perda da GAN é:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x \sim p_{data}(x)}\left[logD(x)\right] + \mathbb{E}_{z \sim p_z(z)}\left[log(1 - D(G(z)))\right]$$

Fig 3 - GAN Loss

DCGAN é uma extensão direta do GAN descrito acima, exceto que ele usa explicitamente camadas convolucionais e convolucionais-transpostas no discriminador e gerador, respectivamente. Ele foi descrito pela primeira vez por Radford et. al. no artigo "Aprendizado de Representação Não Supervisionado com Redes Adversárias Generativas Convolucionais Profundas".

Resultados:

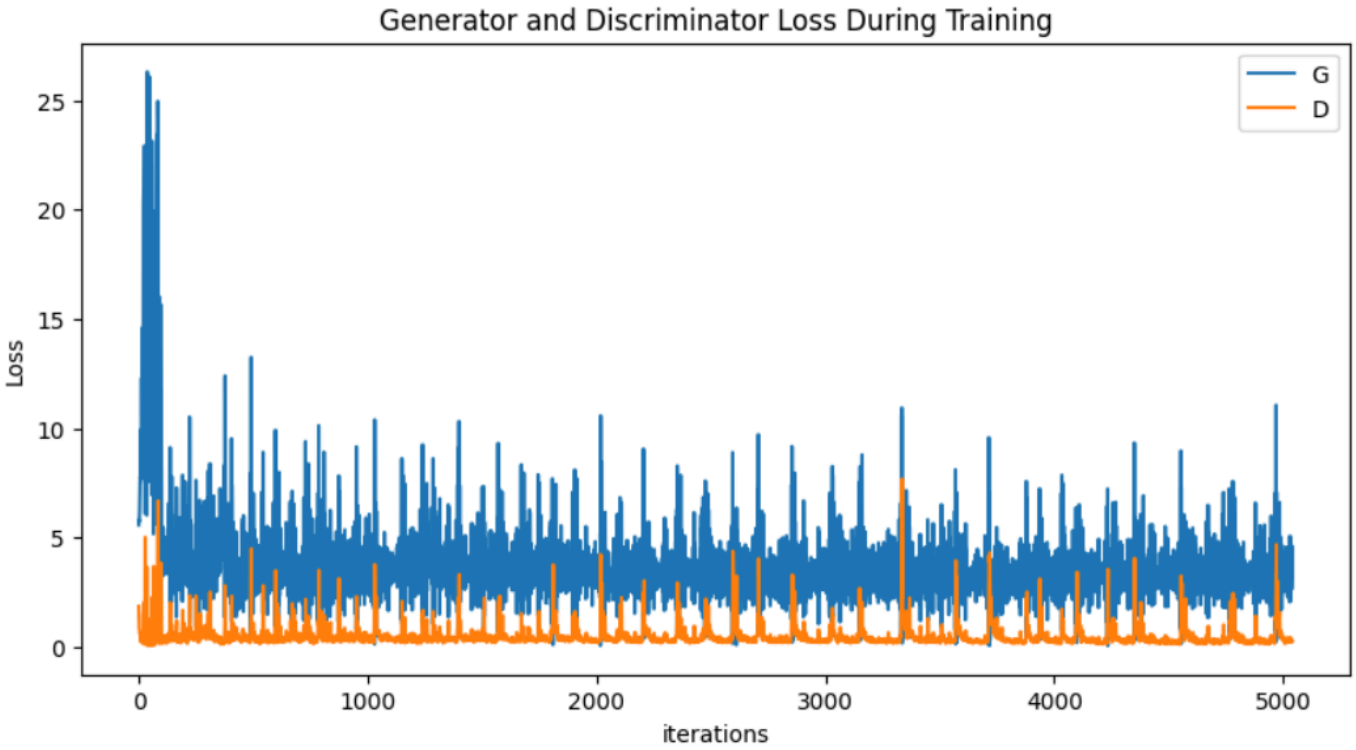

Fig 4 - Perda do gerador e discriminador

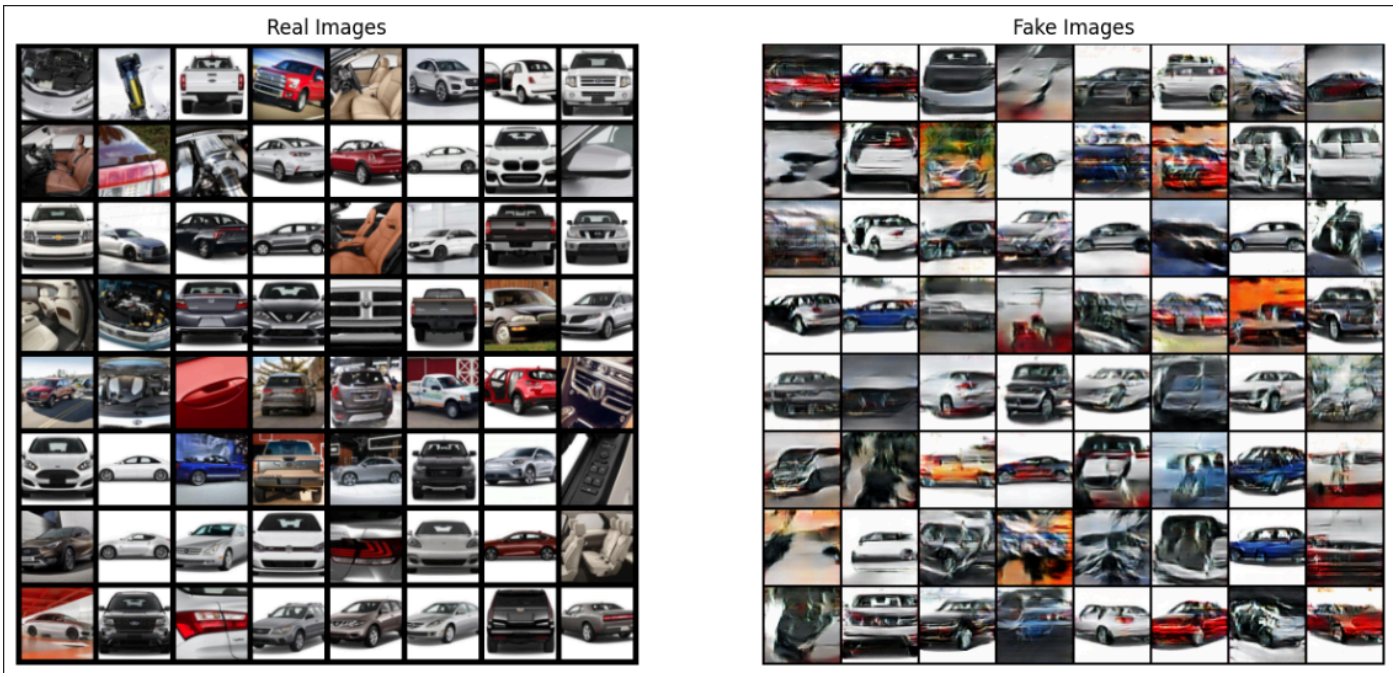

Fig 5 - Comparação Real x Gerado

O resultado final demonstrou deformidades e algumas imagens até com uma boa qualidade visual, posso propor baseado em leituras e instruções oferecidas pelo próprio Pytorch sobre o assunto que eu posso tentar alterar a arquitetura para por exemplo, StyleGAN, Progressive Growing of GANs, no

gerador adicionar um dropout para reduzir overfitting e aumentar a generalização do modelo, ajusta os hiperparâmetros como o batch, testar valores maiores e observar a estabilidade do modelo, podemos testar aumentando o número de épocas ou manipular o ruído.

Existe uma transformação dos dados que ele insere rotações e aumenta consequentemente a quantidade de dados possibilitando um modelo mais detalhado na disputa entre o gerador e discriminador, pode-se na inicialização utilizar autoencoders para acelerar a convergência do modelo.

Próximos passos:

Como vejo em muitos feeds hoje sobre inteligência artificial podemos usar o DCGAN para gerar sequências de imagens como vídeo, aplicar estilo de uma imagem em outra como por exemplo usar as listras de uma zebra e por em uma imagem de cavalo.

Existe um paper sobre espaço latente LAG (Generative Adversarial Networks) o qual o foco está no uso de técnicas adversariais no espaço latente, em vez de diretamente no espaço de amostras observáveis (como imagens ou dados de entrada). Isso significa que o discriminador e o gerador estão competindo em um espaço mais abstrato, onde a representação dos dados pode ser manipulada de forma mais eficiente ou controlada.

Referências

https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html

https://www.kaggle.com/code/eduardomd/dcgan-pytorch/edit

https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf