MLDS

Hw4 Report

R05725004 資管碩二 曾千蕙

1. Model description

   文字使用 one-hot encoding, 頭髮顏色與眼睛顏色共有 23 種，故將文字 encode 成 23 維的向量

   模型架構參考 https://github.com/pytorch/examples/blob/master/dcgan/main.py

   (1) Discriminator:

```
Discriminator(
  (projection): Sequential(
    (0): Linear(in_features=23, out_features=64)
    (1): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True)
    (2): LeakyReLU(0.2, inplace)
  )
  (netD_1): Sequential(
    (0): Conv2d (3, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (1): LeakyReLU(0.2, inplace)
    (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)
    (4): LeakyReLU(0.2, inplace)
    (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)
    (7): LeakyReLU(0.2, inplace)
    (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)
    (10): LeakyReLU(0.2, inplace)
  )
  (netD_2): Sequential(
    (0): Conv2d (576, 1, kernel_size=(4, 4), stride=(1, 1), bias=False)
    (1): Sigmoid()
  )
)
```

   - Projection layer: 將 one-hot encoding 經過一層 linear 層變成維度 64
   - netD_1：將圖片經過 convolution 以及 batch normalization
   - netD_2：將經過 netD_1 的圖片向量以及 projection 後的文字向量接起來後再經過此層 convolution, 並用 sigmoid 得到 0~1 的分數

(2) Generator:

```
Generator(
  (projection): Sequential(
    (0): Linear(in_features=23, out_features=64)
    (1): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True)
    (2): LeakyReLU(0.2, inplace)
  )
  (netG): Sequential(
    (0): ConvTranspose2d (164, 512, kernel_size=(4, 4), stride=(1, 1), bias=False)
    (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True)
    (2): ReLU(inplace)
    (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)
    (5): ReLU(inplace)
    (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)
    (8): ReLU(inplace)
    (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)
    (11): ReLU(inplace)
    (12): ConvTranspose2d (64, 3, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (13): Tanh()
  )
)
```

- Projection layer: 將 one-hot encoding 經過一層 linear 層變成維度 64
- netG: 將文字 vector 跟 noise 向量串接起來後，丟進此層做 deconvolution 以及 batch normalization 生成圖片

2. How do you improve your performance

參考 https://github.com/aelnouby/Text-to-Image-Synthesis/blob/master/trainer.py

在 generator 的 loss function 中加入 (1) 真實圖片與生成圖片的 L1 loss 以及(2) 真實圖片以及生成圖片經過 discriminator 的 convolution 後的向量的 L2 loss, 試圖拉近在一樣的文字 condition 下的生成圖片與真實圖片的距離，不過似乎幫助不大

原 loss function	修改後 loss function
	
	

3. Experiment settings and observation

發現訓練的 iteration 數越多，雖然比較有機會得到正常一點的圖片，但一樣的條件下越容易訓練出非常類似的圖片，推測是 generator 特別去記憶怎樣的圖片 loss 較低，導致於容易有類似甚至一樣的結果

約 150000 iterations	約 170000 iterations