NYCU DLP Lab5

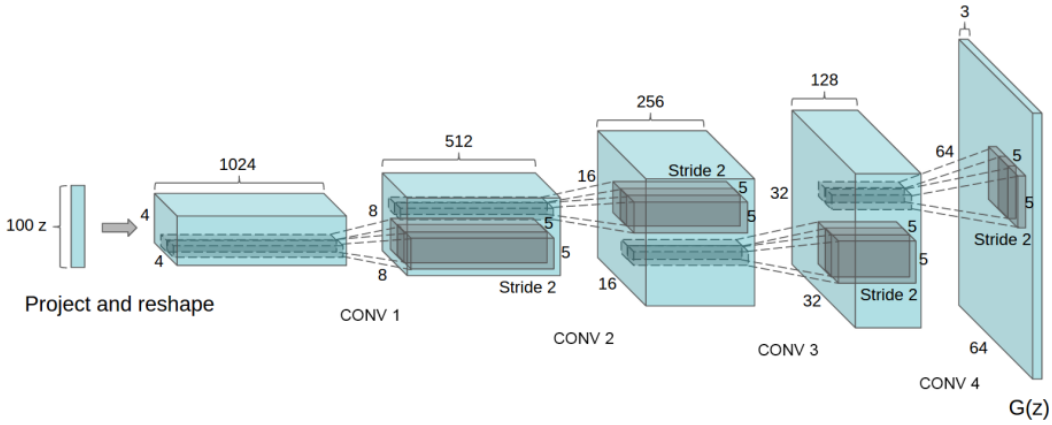
曾信彥

1. Report
   a. Introduction

      此次 Lab 使用 conditional GAN 來生成多標籤的圖片，training data 是 ICLEVR 中的一堆彩色方塊，共 3 個形狀, 8 種顏色，因此 condition 設成一個 24D 的 one-hot vector，test 時會給予 condition，每次最多三個，要生成特定的圖片，所以我們透過訓練 Conditional GAN 來實作這次的 LAB

   b. Implementation details

      Model Architecture, DCGAN:



      Generator:

```
Generator(
  (conditionExpand): Sequential(
    (0): Linear(in_features=24, out_features=200, bias=True)
    (1): ReLU()
  )
  (convG1): Sequential(
    (0): ConvTranspose2d(300, 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)
  )
  (convG2): Sequential(
    (0): ConvTranspose2d(512, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
  )
  (convG3): Sequential(
    (0): ConvTranspose2d(256, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
  )
  (convG4): Sequential(
    (0): ConvTranspose2d(128, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
  )
  (convG5): ConvTranspose2d(64, 3, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
  (tanh): Tanh()
)
```

      Generator 會把預設的 condition expand 再 concat noise 再經過多層的 transpose CNN, 最後輸出一張 Fake Image with size 64*64

Discriminator:

```
Discriminator(
  (conditionExpand): Sequential(
    (0): Linear(in_features=24, out_features=4096, bias=True)
    (1): LeakyReLU(negative_slope=0.01)
  )
  (convD1): Sequential(
    (0): Conv2d(4, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): LeakyReLU(negative_slope=0.2, inplace=True)
  )
  (convD2): Sequential(
    (0): Conv2d(64, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): LeakyReLU(negative_slope=0.2, inplace=True)
  )
  (convD3): Sequential(
    (0): Conv2d(128, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): LeakyReLU(negative_slope=0.2, inplace=True)
  )
  (convD4): Sequential(
    (0): Conv2d(256, 512, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): LeakyReLU(negative_slope=0.2, inplace=True)
  )
  (convD5): Conv2d(512, 1, kernel_size=(4, 4), stride=(1, 1), bias=False)
  (sigmoid): Sigmoid()
)
```

Discriminator 的輸入除了 real image 還有 fake image，還會把圖片對應的 condition 一起輸入到 D 裡面，因為 D 原本要輸入 image，但 image 和 condition 的 size 不同，所以先用 fully connected layer 將 24 放大到 4096(64*64)再和 image concat 變成 4 個 channel 再輸入進 D，之後經過五層的 CNN 再經 sigmoid 輸出 0~1 代表他是 real image 還是 fake image 的機率

Loss function:

Loss 的部分跟原本 DCGAN 裡面一樣使用 nn.BCELoss()去計算照片真假預測的失誤

Hyperparameters:

Batch size: 32

Lr: 0.0002

Epochs: 300

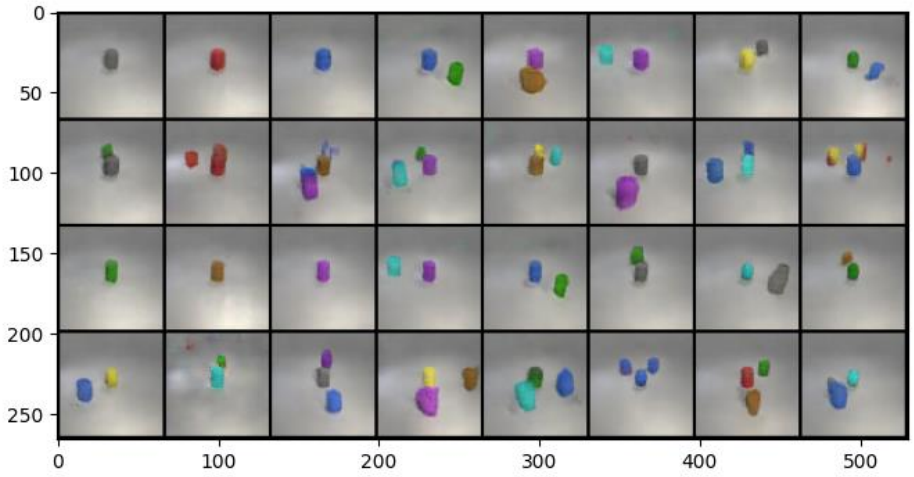
Optimizer: Adam

Image size: 64*64

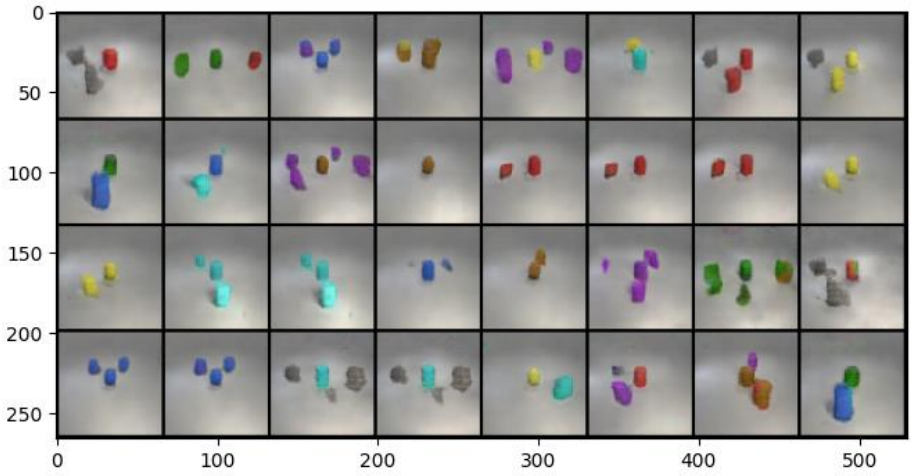
Noise size: 100

Condition size: 200

2. Show your results based on the testing data.
   a. test.json



   b. new_test.json



   c. Scoring

```
Score: 0.77

Score: 0.73
```

   On test.json get 0.77

   On new_test.json get 0.73

Discuss the results of different models architectures:

在訓練 DCGAN 的時候，一開始的效果很糟發現 discriminator 的 loss 掉到 0 然後就訓練不下去，為了要成功收斂，就要小心的調參數，learning rate 不能太大(0.0002 算是比較好的狀況，可能太大會導致 gradient vanishing)，epoch 不能設太小(GAN 是經由迭代彼此互相訓練，要經過足夠多的 epoch 之後才會有好結果)，Activation 和 Architecture 則是參照 DCGAN 的論文 Unsupervised Representation Learning With Deep Convolutional Generative Adversarial Networks, Generator 用 ReLu()為 layer activation 和 Tanh()為 output activation. Discriminator 則是用 LeakyReLu()和 sigmoid().

實驗目前的結果是，使用原本 DCGAN 使用的 BCELoss 效果是最好的，比起其他的 loss 衡量辦法，還是先用 BCELoss 在 DCGAN 上或許才是個好選擇