Lab5 Report

1. Introduction

實作 cGAN，用 generator 產生圖片 ，並用 discriminator 分辨圖片

真假及 label

2. Implementation details

choice of cGAN: 使用 cDCGAN

model architectures:

```
Generator(
  (conditionExpand): Sequential(
    (0): Linear(in_features=24, out_features=200, bias=True)
    (1): ReLU()
  )
  (convT1): Sequential(
    (0): ConvTranspose2d(300, 512, kernel_size=(4, 4), stride=(2, 2))
    (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
  )
  (convT2): Sequential(
    (0): ConvTranspose2d(512, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
  )
  (convT3): Sequential(
    (0): ConvTranspose2d(256, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
  )
  (convT4): Sequential(
    (0): ConvTranspose2d(128, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
  )
  (convT5): ConvTranspose2d(64, 3, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
  (tanh): Tanh()
)
```

```
Discriminator(
  (conditionExpand): Sequential(
    (0): Linear(in_features=24, out_features=12288, bias=True)
    (1): LeakyReLU(negative_slope=0.01)
  )
  (conv1): Sequential(
    (0): Conv2d(6, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): LeakyReLU(negative_slope=0.01)
  )
  (conv2): Sequential(
    (0): Conv2d(64, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): LeakyReLU(negative_slope=0.01)
  )
  (conv3): Sequential(
    (0): Conv2d(128, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): LeakyReLU(negative_slope=0.01)
  )
  (conv4): Sequential(
    (0): Conv2d(256, 512, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): LeakyReLU(negative_slope=0.01)
  )
  (conv5): Conv2d(512, 1, kernel_size=(4, 4), stride=(1, 1))
  (sigmoid): Sigmoid()
)
```

Generator 的 input 是 100 維的 noise(高斯分布)和 24 維的 label，首先先將 24 維擴充到 200 維再和 noise concat 起來，餵進 5 層 convolution，前 4 層包含 BN 和 ReLU，最後一層接 tanh 輸出 fake image。

Discriminator 的 input 是 real image 和 label，label 先擴充成 3*64*64 維 再跟 real image concat 變成(6, 64, 64)，其餘跟 generator 類似，最後一層使用 sigmoid

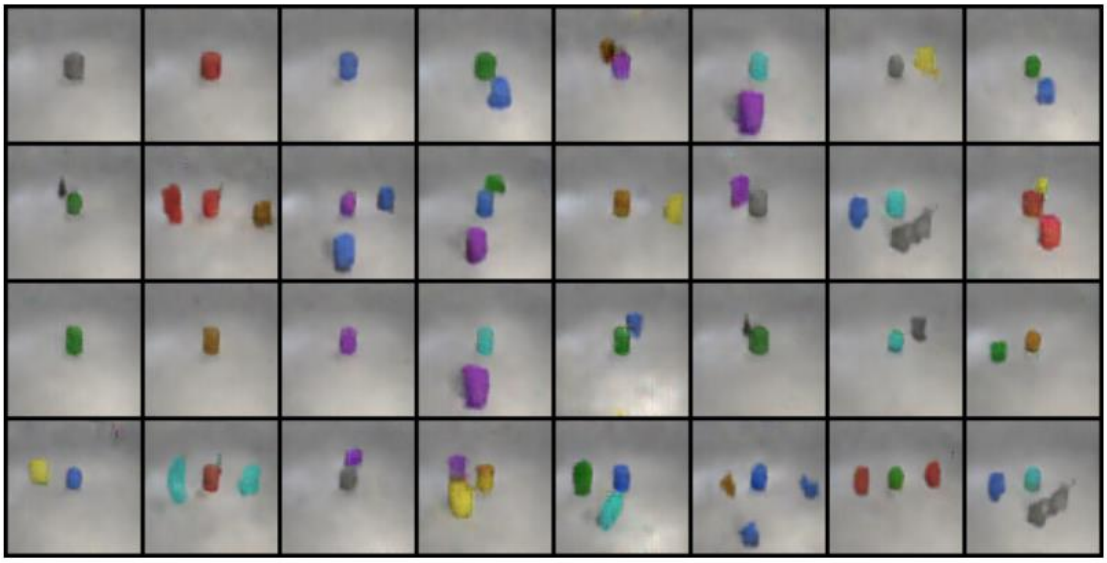
loss functions: 2 個 model 都使用 nn.BCELoss()

hyperparameters:

```
device = torch.device('cuda:0' if torch.cuda.is_available() else 'cpu')
batch_size = 64
z_dim = 100
c_dim = 200
image_shape = (64, 64, 3)
lr = 0.0002
epochs = 200
```

3. Results and discussion

```
testing score: 0.67
```



在這次 lab 中我使用的 condition 是 label，有看到其他的作法是使用其他實際的圖片作為 condition，理論上應該會比 label 更明確的資訊，因為 1 個 label 可能有好幾種圖片的樣子