Lab 4: InfoGAN

系所：多媒體工程研究所

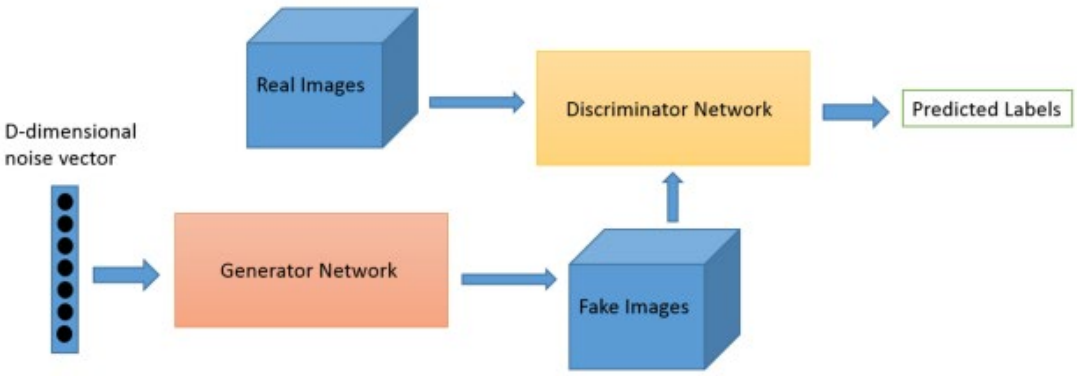
學號：0756616

姓名：周冠伶

1. Introduction

Generative Adversarial Network(GAN)包含 Generator Network 和 Discriminator Network，分別用於偽造假資料和測試資料，透過兩個網路彼此互相訓練，使結果能夠模擬出真實資料的分佈。本次實作重點在於學習如何使用 GAN 的架構進行分類，並於 MNIST 上使用實作的模型測試結果。

InfoGAN 和 DCGAN 都是 GAN 的衍伸模型，其中兩者最大的差異在於 InfoGAN 會將輸入的噪聲放入 Q 模型中，計算一組 Cross Entropy Loss 作為輸出，並控制特定輸入與輸出的依賴關係。本次實作將會以 InfoGAN 為主要架構，並配合使用 DCGAN 的生成模型與辨識模型。



2. Implementation details

A. Describe generator and discriminator architectures

```
class Generator(torch.nn.Module):
    def __init__(self):
        super(Generator, self).__init__()
        self.main = torch.nn.Sequential(
            # input is Z, going into a convolution
            torch.nn.ConvTranspose2d(74, 512, 4, 1, 0, bias=False),
            torch.nn.BatchNorm2d(512),
            torch.nn.ReLU(True),
            torch.nn.ConvTranspose2d(512, 256, 4, 2, 1, bias=False),
            torch.nn.BatchNorm2d(256),
            torch.nn.ReLU(True),
            torch.nn.ConvTranspose2d(256, 128, 4, 2, 1, bias=False),
            torch.nn.BatchNorm2d(128),
            torch.nn.ReLU(True),
            # change size
            torch.nn.ConvTranspose2d(128, 1, 4, 2, 3, bias=False),
            #torch.nn.BatchNorm2d(64),
            #torch.nn.ReLU(True),
            #torch.nn.ConvTranspose2d(64, 1, 4, 2, 1, bias=False),
            torch.nn.Tanh())

    def forward(self, input):
        output = torch.nn.parallel.data_parallel(self.main, input, range(1))
        return output
```

生成模型：修改 InfoGAN 反卷積、BatchNormal 的參數做修改，並於每次反卷積與 BatchNomal 後加入激勵函式 ReLU；修改最後輸出層的 Sigmoid 激勵函式為 Tanh。

```
class D(torch.nn.Module):
    def __init__(self):
        super(D, self).__init__()

        self.main = torch.nn.Sequential(
            torch.nn.Conv2d(1024, 1, 1), torch.nn.Sigmoid())

    def forward(self, x):
        output = self.main(x).view(-1, 1)
        return output
```

辨識模型：將輸入做二維卷積與激勵函式 Sigmoid 後攤平張量。

```python
class FE(torch.nn.Module):
    ''' front end part of discriminator and Q'''

    def __init__(self):
        super(FE, self).__init__()

        self.main = torch.nn.Sequential(
            torch.nn.Conv2d(1, 64, 4, 2, 1),
            torch.nn.LeakyReLU(0.1, inplace=True),
            torch.nn.Conv2d(64, 128, 4, 2, 1, bias=False),
            torch.nn.BatchNorm2d(128),
            torch.nn.LeakyReLU(0.1, inplace=True),
            torch.nn.Conv2d(128, 1024, 7, bias=False),
            torch.nn.BatchNorm2d(1024),
            torch.nn.LeakyReLU(0.1, inplace=True),
        )

    def forward(self, x):
        output = self.main(x)
        return output
```

FrondEnd 模型：使用於辨識模型前，用於資料前處理。該模型會對資料進行卷積、BatchNormal 與激勵函式 LeakyReLU。

B. Specify the hyper-parameters and setting

BATCH_SIZE = 100

EPOCH = 50

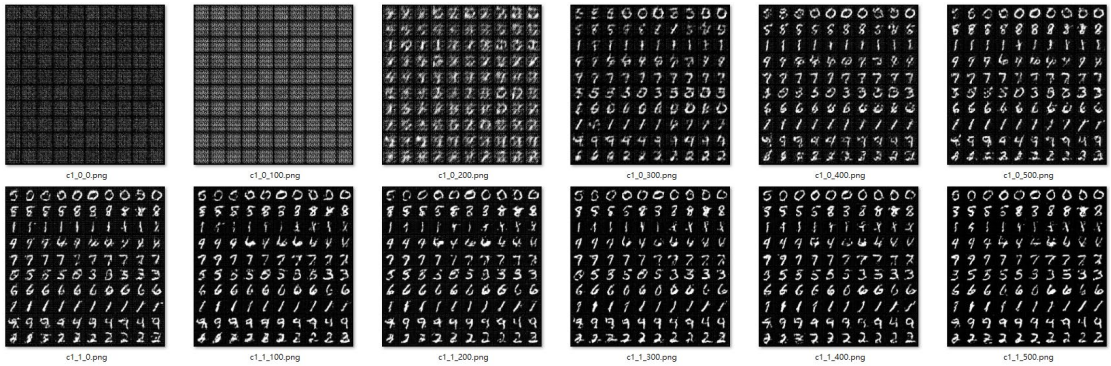
Optimizer Discriminator/Generator LEARNING_RATE = 0.0002/0.001

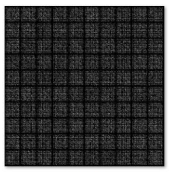
模型參數參照 2-A

3. Results and discussion

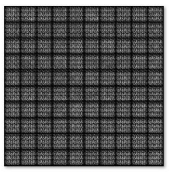
A. Results of all MNIST numbers

下圖依序為寬度特徵(c1)與螺旋特徵(c2)於訓練過程中前 12 個 Epoch、最終成果圖，以及測試生成數字 0~9 的結果圖(數字對應訓練標籤順序)：

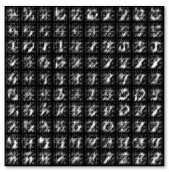




c2_0_0.png



c2_0_100.png



c2_0_200.png

c2_0_300.png

c2_0_400.png

c2_0_500.png

c2_1_0.png

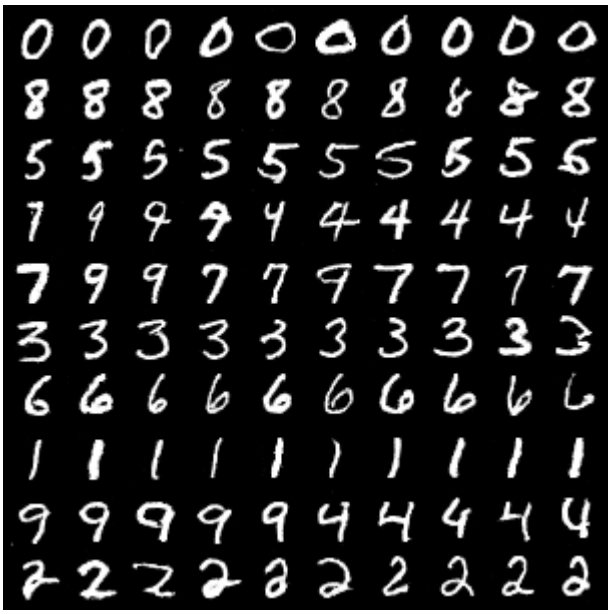
c2_1_100.png

c2_1_200.png

c2_1_300.png

c2_1_400.png

c2_1_500.png



TestC1_0.png

TestC1_1.png

TestC1_2.png

TestC1_3.png

TestC1_4.png

TestC1_5.png

TestC1_6.png

TestC1_7.png

TestC1_8.png

TestC1_9.png

TestC2_0.png

TestC2_1.png

TestC2_2.png

TestC2_3.png

TestC2_4.png

TestC2_5.png

TestC2_6.png

TestC2_7.png

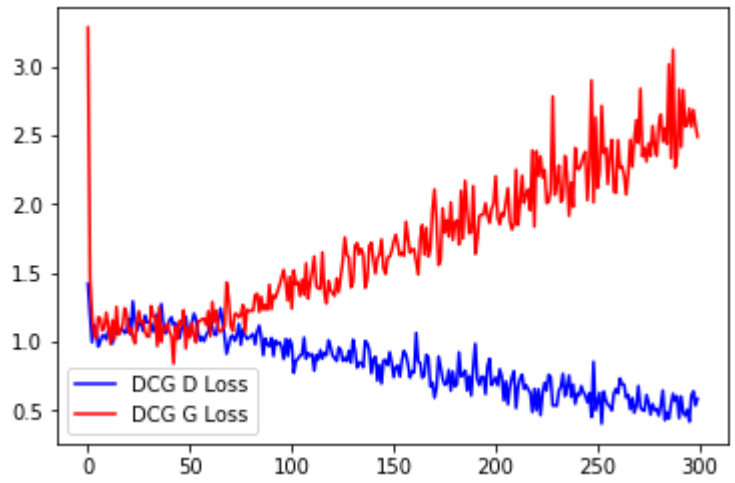
TestC2_8.png

TestC2_9.png

B. Discuss training process the results of the loss curves

- Discriminator loss 與 Generator loss：

  在進行 Training 時，兩者 Loss 在初期會差異較大、並隨著 Epoch 的進行而開始互相接近；但是如果訓練 Epoch 過多時，訓練結果會開始變差、兩者 Loss 會開始分離(如下圖，Epoch>80 後兩者開始分離)。



- 模型參數修改：

  在修改 DCGAN 的模型、以加入 InfoGAN 的架構時，可以選擇透過修改(1)層數(2)參數(如：Channel、Kernel Size 等)(3)使用的函式(如：激勵函式)等方式進行。

  在實作過程中，最初是選擇使用 Linear 來改變維度，預期能夠與原本的模型對齊輸入與輸出，但是輸入的圖片是二維的，該函式並不適用此狀況；最後透過修改(1)層數和(2)參數來達到目的。

- Condition：

  在輸入噪聲中，另外加入用於控制生成數字的 Condition 標籤，使資料在經過模型後的輸出可以被控制，此過程與 Mutual Information 有關。與生成假資料的過程簡單(數字數據)-複雜(數字圖)-簡單相反，該過程為複雜-簡單-複雜。

  其中 Condition 標籤使用 One-Hot 形式呈現，透過 10 個 1*10 的 Vector 來表示數字 0~9。One-Hot Vector 與數字 0~9 非使用數字對齊，而是依據模型訓練的標籤。另外還有計算 c1 與 c2 表示寬度特徵與螺旋特徵。