Lab6 InfoGAN

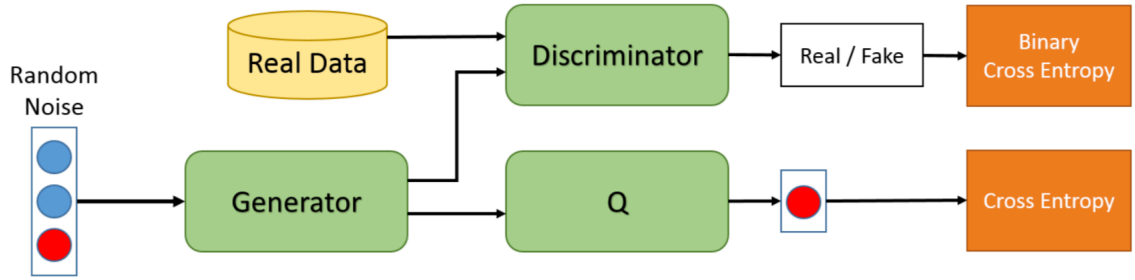
0516054 劉雨恩

1. Introduction

本次實驗目的為架構 infoGAN 神經網路，並使用 MNIST 資料集來訓練網路。採用 adversarial loss 來生成逼真圖像，和透過 maximize mutual information 來學習 disentangled representations。目標為 generator network 在 input 為固定數字(0~9)、不同 noise，以及固定 noise、不同數字的兩種情況下產生 10x10 個圖像。

2. Experiment setups

A. How you implement InfoGAN



I. Adversarial loss

```
############################
# (1) Update D network
############################

optimD.zero_grad()
netD.zero_grad()

# train with real
D_real_out = netDQFront(batch_x)
D_real_out = netD(D_real_out)
T_pro_part += D_real_out.sum().item()
D_real_loss = params.BCE(D_real_out, real_label)

# train with fake
G_out = netG(noise)
D_fake_out = netDQFront(G_out)
D_fake_out = netD(D_fake_out)
F1_pro_part += D_fake_out.sum().item()
D_fake_loss = params.BCE(D_fake_out, fake_label)

# update
D_loss = D_real_loss + D_fake_loss
D_loss_part += D_loss.item()
D_loss.backward(retain_graph=True)
optimD.step()
```

Discriminator net 訓練分為兩次，分別放入由 MNIST train dataset 而來的 real data 和由 Generator net 生成的 fake data。兩次 output 都會和各自的 real_label 或 fake_label 來計算 binary cross entropy 的 loss，相加便是 adversarial loss。最後再 backward 計算 gradient 對 Discriminator net 進行更新。

而 optimD 的 parameters 由 Discriminator 和 Q 共用的 net 和 Discriminator 的 net 而來。

```
optimD = t.optim.Adam([{'params': netDQFront.parameters()}, {'params': netD.parameters()}],
                      lr=params.LR_DIS)
```

II. Maximizing mutual information

```
############################
# (2) Update G & Q network
############################
optimG.zero_grad()
netG.zero_grad()
netQ.zero_grad()

# train with D
D_G_out = netDQFront(G_out)
D_G_out = netD(D_G_out)
D_G_loss = params.BCE(D_G_out, real_label)

# train with Q
Q_out = netDQFront(G_out).view(-1, 8192)
Q_out = netQ(Q_out)
Q_loss = params.CE(Q_out, cate_noise)
Q_loss_part += Q_loss.item()

## Q accuracy
Q_out_index = t.max(Q_out, 1)[1]
Q_hat = sum(Q_out_index == cate_noise).item()
Q_acc_part += Q_hat

# update
G_loss = D_G_loss + Q_loss
G_loss_part += G_loss.item()

G_loss = 10*D_G_loss + 100*Q_loss
G_loss.backward()
optimG.step()
```

將 Generator net 生成的 fake data 放入 Q net 後，將生成的 output 和當初放入 Generator net 的 one hot vector 代表的 label 計算 Cross Entropy Loss，便是 mutual information loss。

將 Generator net 生成的 fake data 再次放入 Discriminator net，但這次出來的 ouput 是和 real_label 計算 binary cross entropy 的 loss。

最後將上述兩種 loss 相加再 backward 計算 gradient 對 Generator net 和 Q net 進行更新。

而 optimG 的 parameters 由 Generator 的 net 和 Q 的 net 而來。

```
optimG = t.optim.Adam([{'params': netG.parameters()}, {'params': netQ.parameters()}],
                      lr=params.LR_GEN_Q)
```

III. How you generate fixed noise and images

- Random Noise

放入 Generator 的 noise 是由兩種不同的 noise——standard normal distribution 和 label 的 one hot vector concat 而來。

1. 54-D continuous noise drawn from standard normal distribution.

```
def create_continuous_noise(size):
    out = np.random.standard_normal(size=(size, params.Z_SIZE))
    out = t.from_numpy(out).cuda()
    return out
```

從 standard normal distribution 中 sample 出 size 為 batch_size X z_size 大小的 noise。

2. 10-D discrete one hot vector.

```
def create_categorical_noise(size, fixed=False):
    noise = []
    numbers = []
    for i in range(size):  # one-hot vector
        noise_i = np.zeros(params.C_SIZE, dtype=int)
        if(fixed==False):
            index = np.random.randint(0, params.C_SIZE)
        else:
            index = i
        noise_i[index] = 1
        noise.append(noise_i)
        numbers.append(index)
    noise = t.cuda.DoubleTensor(noise)
    numbers = t.cuda.LongTensor(numbers)
    return noise, numbers
```

先生成 size 為 c_size 大小全是 0 的矩陣，再隨機生成 0~9 之間的數字，並將隨機生成的數字作為矩陣元素變成 1 的 index。

Ex.

step1. a = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

step2. random 出來一個數字為 2

step3. a[2] = 1

- Images

```
def test(netG, noise, epoch_i, batch_i):
    netG.eval()

    numbers, _ = dl.create_categorical_noise(params.C_SIZE, True)
    outGs = []
    with t.no_grad():
        for i in range(10):  # different noise
            for j in range(10):  # different number
                noise_cat = t.cat((noise[i], numbers[j]), dim=0).view(1, 64, 1, 1)
                noise_cat = noise_cat.type(t.FloatTensor).cuda()
                outG = netG(noise_cat)
                outGs.append(outG)

    images = t.cat(outGs, dim=0)
    vutils.save_image(images,
                      '%s/testImg_%d_%d.png' % (params.RESULT_FILE, epoch_i, batch_i),
                      normalize=True,
                      nrow=10)

    netG.train()
```

Step1. 生成兩種noise：random drawn from standard normal distribution(size: 10x54)和label 0~9的one hot vector(size: 10x10)

Step2. 設立兩個迴圈把noise放入Generator net：可得到不同noise同一個number和不同number同一個noise的情況

Step3. vutils.save_image()：將所有得到的Generator net output利用vutils.save_image()生成10張為一排的圖像

IV. Parameters

- batch size: 64

- learning rate for the discriminator: 1e-4

- learning rate for the generator and Q: 1e-3
- z_size (size of noise from standard normal distribution): 54
- c_size (size of meaningful codes): 10
- total epoch: 80
- loss function: torch.nn.BCELoss()

  & torch.nn.CrossEntropyLoss()
- optimizer function: torch.optim.Adam()
- image size: 64
- real_label: 1
- fake_label: 0

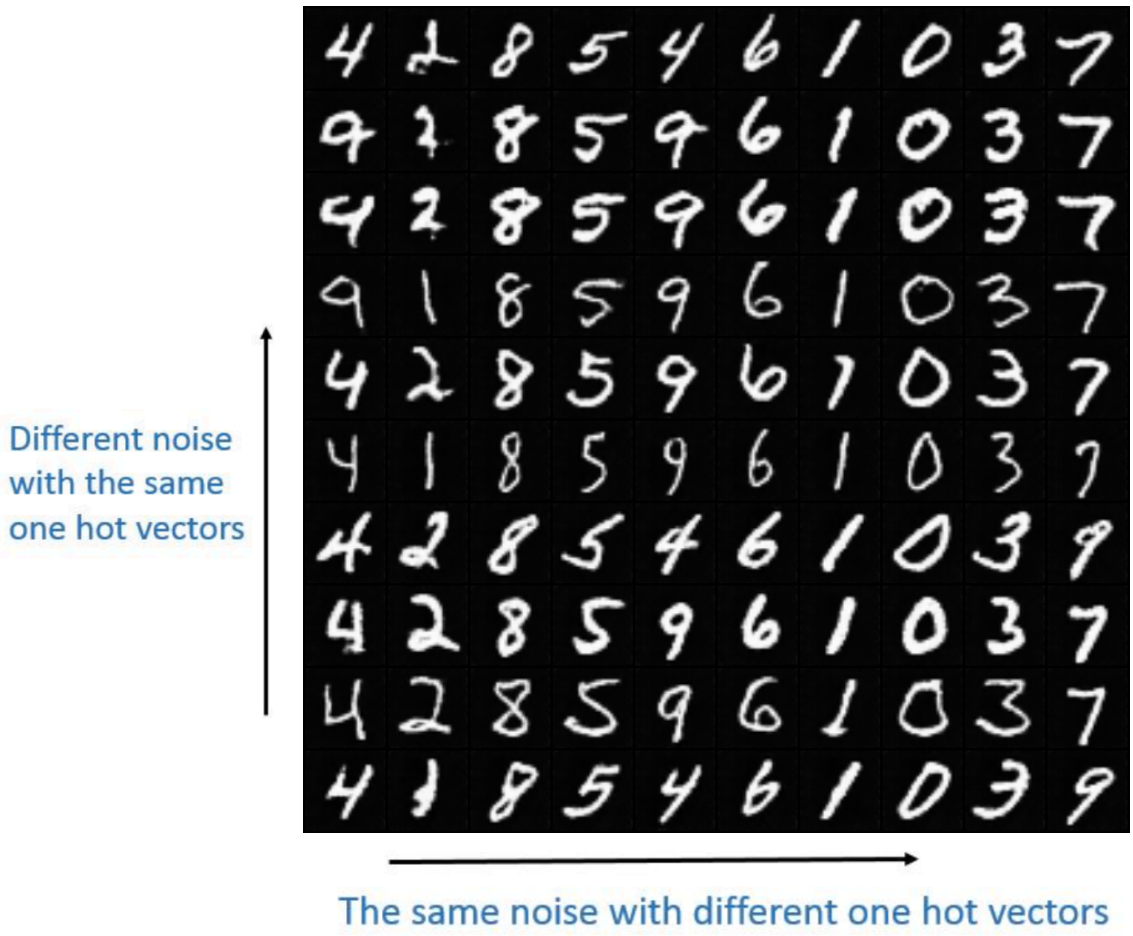
B. Which loss function of generator you used

$$\mathcal{L}_{\mathrm{G}} = E_{x\sim p_g}[-\log D(x)] + \mathrm{L}_{\mathrm{I}}(G, Q)$$

所用的 Generator loss 是將 Generator net 生成的 fake data 放入 Discriminator net，出來的 ouput 和 real_label 計算 binary cross entropy 而來的 loss。並非如另一個式子是 Discriminator loss 的相反，也就是直接取 Discriminator loss 的負數。

3. Results

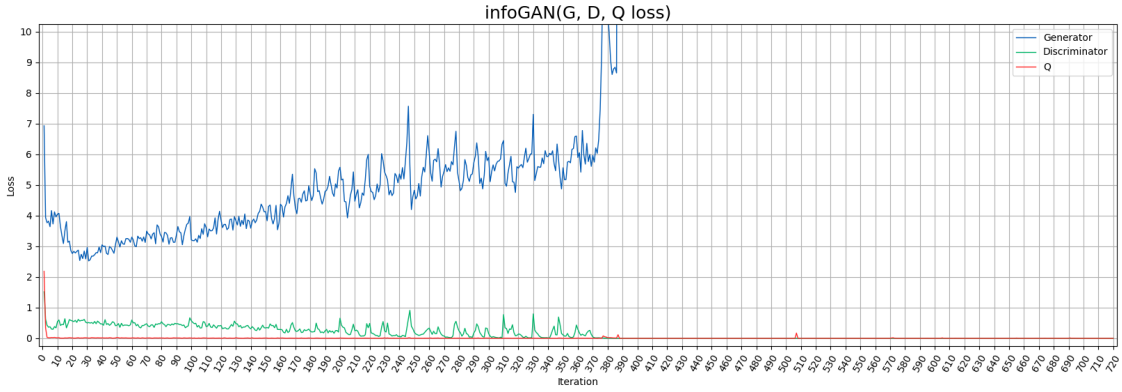
A. Results of your samples



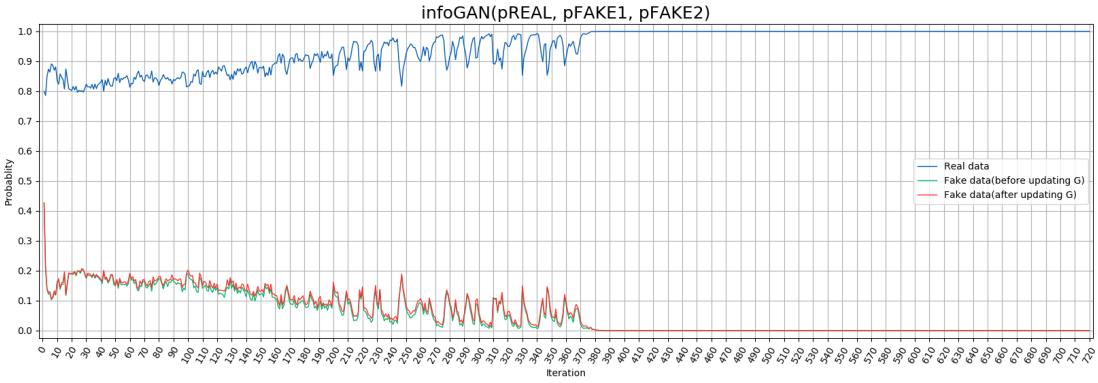
從左而右的number為4, 2, 8, 5, 9, 6, 1, 0, 3, 7。

B. Training loss curves (every 100 batch steps)

I. Loss of the generator, the discriminator, and the Q.



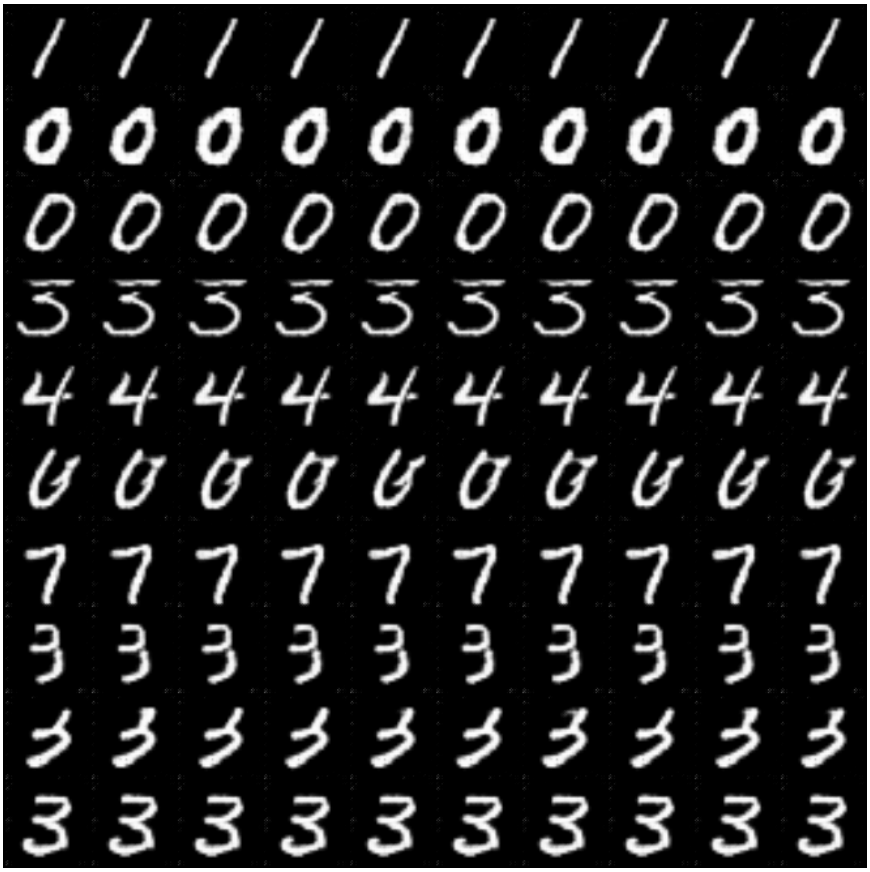
II. Probability of real data, fake data before updating G and after updating G.



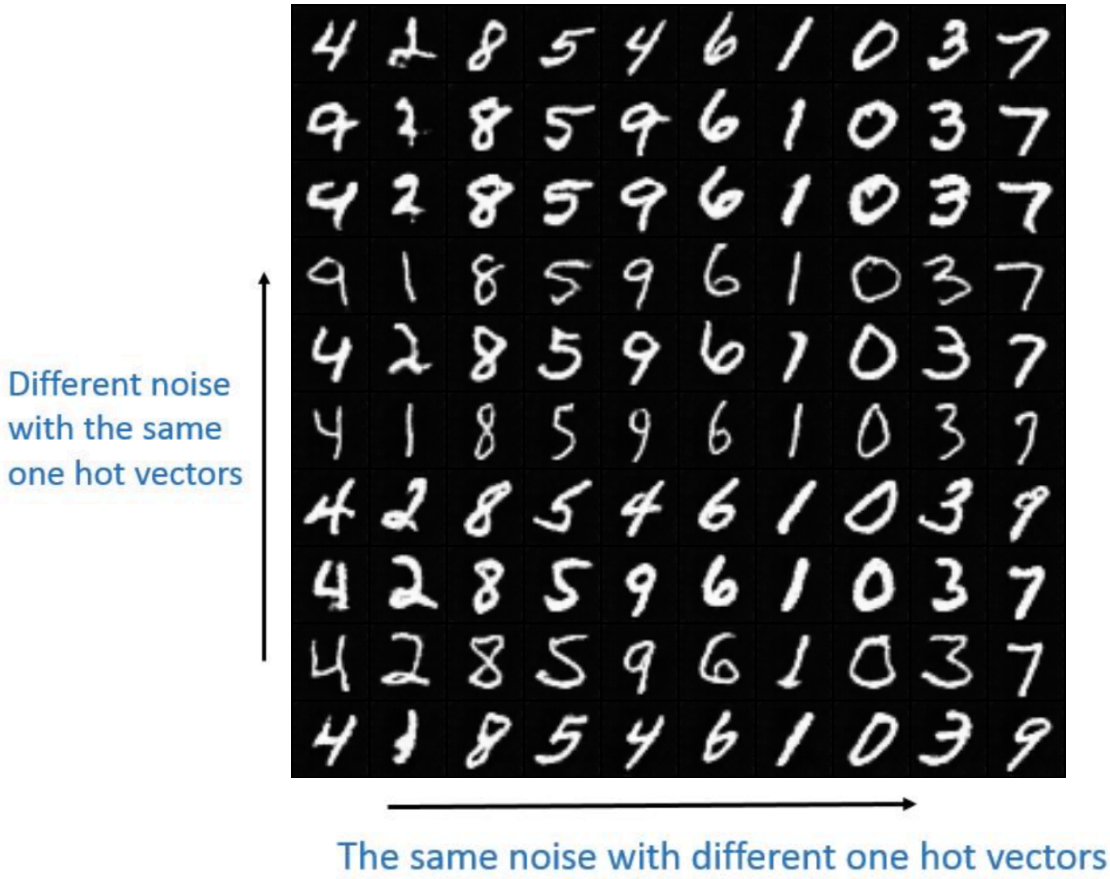
4. Discussion

- 生成錯誤圖像

實驗過程中，曾出現以下的圖像。原本以為是生成 10x10 圖片做兩層迴圈的時候，index 放相反了；但後來發現 Q 的 accuracy 並不高，也就是說 Q net 沒有學到 one hot vector label 的 noise。後來將 generator loss 中 Q loss 的比重提高，Q 的 accuracy 就能在訓練過程中變為 1，而圖片也成功生成正確格式。



- labels & true numbers



net 無法學習 one hot vector label 應該對應的那個數字，所以最後由 Generator 得到的數字會和 label 的不一致。

label	0	1	2	3	4	5	6	7	8	9
number	4	2	8	5	9	6	1	0	3	7