Lab 4 InfoGAN

0851915 徐玉山

A. Introduction (10%)

DCGAN can generate hand-written digit image but it can't control generator to generate image
with specific digit number. To solve that situation, need use InfoGAN to learn condition from noise.
Lab's requirements as follows:
1. Modify the generator and the discriminator in InfoGAN (https://github.com/pianomania/infoGAN-pytorch)

2. Adopt traditional generator and discriminator loss. Maximize the mutual information between generated images and discrete one-hot vector

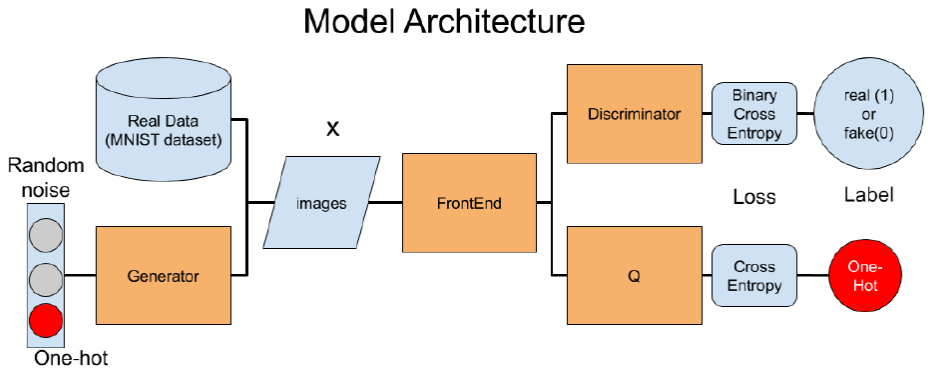
3. Show the generated images of the specific number

4. Plot the loss curves of the generator and the discriminator while training

目標: 要用infogan訓練出一個generator可以拿 one-hot condition和random noise 做 input 來產生數字圖片。

B. Implementation details (30%)

◆ Describe your generator and discriminator architectures. You must adopt other different model architecture (different channel size, activation function, normalization, layer numbers, etc). Network Q can simply follow the reference code. (20%)



```python
class G(nn.Module):

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

    self.main = nn.Sequential(
      nn.ConvTranspose2d(64, 512, 1, 1, bias=False),
      nn.BatchNorm2d(512),
      nn.ReLU(True),
      nn.ConvTranspose2d(512, 256, 7, 1, bias=False),
      #nn.ConvTranspose2d(1024, 128, 8, 1, bias=False),
      nn.BatchNorm2d(256),
      nn.ReLU(True),
      nn.ConvTranspose2d(256, 64, 4, 2, 1, bias=False),
      #nn.ConvTranspose2d(128, 64, 5, 2, 1, bias=False),
      nn.BatchNorm2d(64),
      nn.ReLU(True),
      nn.ConvTranspose2d(64, 1, 4, 2, 1, bias=False),
      nn.Sigmoid()
    )
```

```python
class G(nn.Module):

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

    self.main = nn.Sequential(
      nn.ConvTranspose2d(64, 2048, 1, 1, bias=False),
      nn.BatchNorm2d(2048),
      nn.ReLU(True),
      nn.ConvTranspose2d(2048, 512, 7, 1, bias=False),
      #nn.ConvTranspose2d(1024, 128, 8, 1, bias=False),
      nn.BatchNorm2d(512),
      nn.ReLU(True),
      nn.ConvTranspose2d(512, 64, 4, 2, 1, bias=False),
      #nn.ConvTranspose2d(128, 64, 5, 2, 1, bias=False),
      nn.BatchNorm2d(64),
      nn.ReLU(True),
      nn.ConvTranspose2d(64, 1, 4, 2, 1, bias=False),
      nn.Sigmoid()
```

我改變了 generative model 裡的 hidden units 數目

一個改得比原本來的多，另外一個比原本來的少，效果上感覺沒什麼差別

◆ Specify the hyperparameters and setting (e.g. channels, learning rate, optimizer, epochs, etc.) (10%)

```
criterionD = nn.BCELoss().cuda()
criterionQ_dis = nn.CrossEntropyLoss().cuda()
criterionQ_con = log_gaussian()

optimD = optim.Adam([{'params':self.FE.parameters()}, {'params'
optimG = optim.Adam([{'params':self.G.parameters()}, {'params':
```

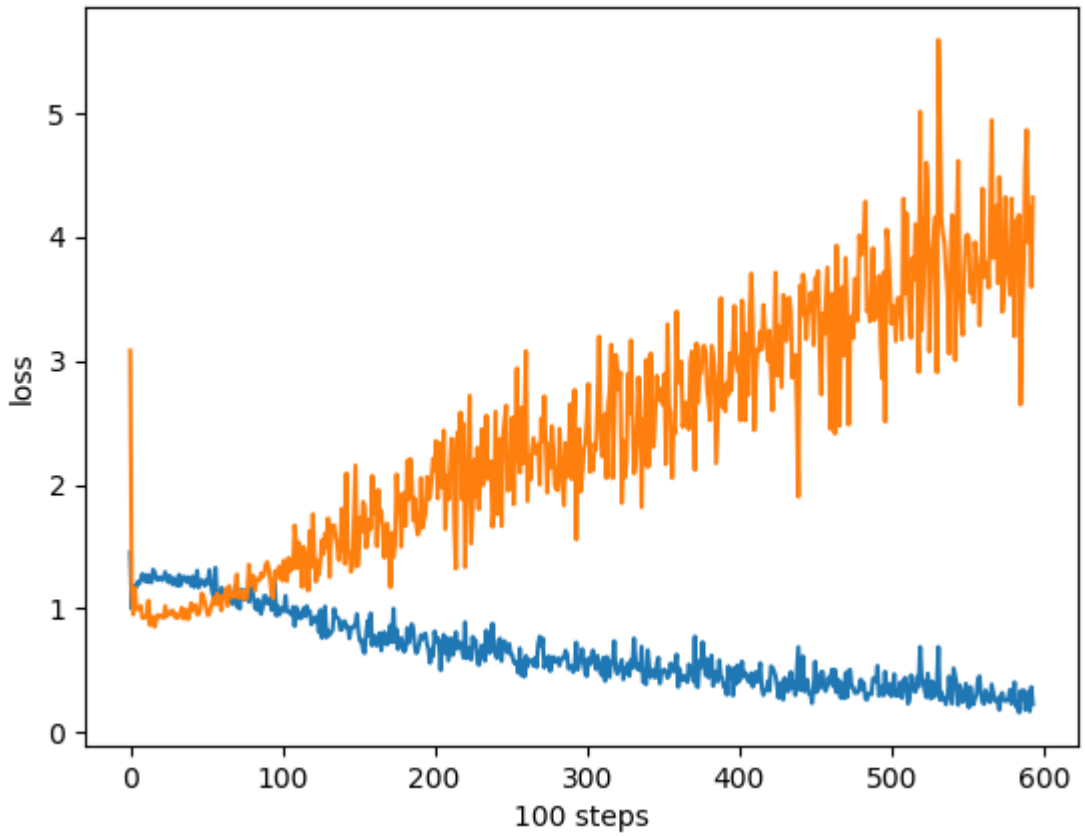
C. Results and discussion (20%)

◆ Show your results of all MNIST numbers (10%)

Generator 用比較少神經元練 100 epoch 之結果

Generator 用比較多神經元練 100 epoch 之結果

◆ Discuss your training process the results of the loss curves (10%)


loss
0
1
2
3
4
5
0
100
200
300
400
500
600
100 steps