深度學習

HW13

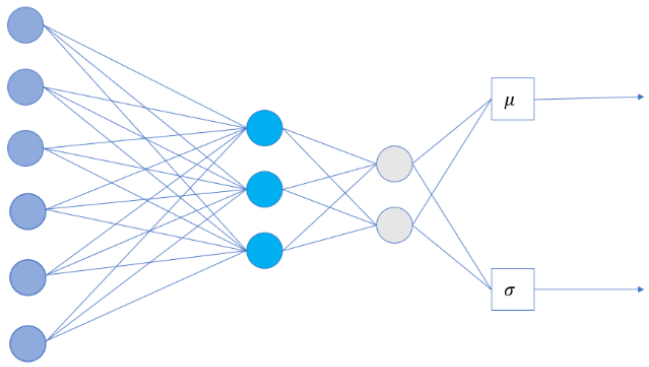
學號 : B103012002

姓名 : 林凡皓

VAEs 和 GANs 在訓練過程中對超參數非常敏感，而且通常需要很多訓練 epochs。為了讓這次練習更加容易完成，我們使用 MNIST 資料及來做練習。MNIST 資料即包含 60000 張訓練圖片與 10000 張測試圖片，每一張圖片都包含一個位於黑色背景中心位置上的白色數字。

一、 FC-VAE Encoder

Fully-connected VAE network encoder 會接收一張圖片並將他 flatten，然後送進三層的 Linear + ReLU layers。我們會使用 hidden dimension representation 和兩個 Linear layer 來預測 posterior μ 和 posterior log variance。架構大致如下



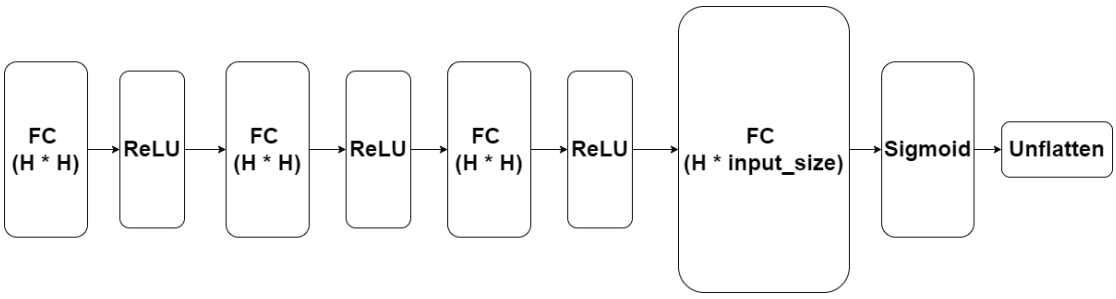
其中中間藍色的 Fully connected layer 會有三個。

- 實現方式：

  利用 nn.Sequential 來將 Flatten、Linear、ReLU 串接起來，經過此 sequential model 後的 shape = (N, hidden_dimention)。在此 Sequential model 後方接上兩個 Linear layer，分別用來將 hidden features 映射到預測 posterior mean 和 posterior log variance，mean 和 log variance 的形狀皆為(N, Z)。

二、 FC-VAE Decoder

Decoder 會用 latent space representation 來還原圖片，其架構如下



- 實現方法：

  有點類似於 encoder 的相反。利用 nn.Sequential 來實現 decoder 架構。該架構會接收 shape = (N, Z)的輸入，並輸出與 encoder 輸入相同形狀。

三、 Reparametrization

在 forward path 中，我們會需要根據 mean 和 variance 來估計 posterior z。一種最簡單的實現方式為根據我們的 mean 和 variance 產生 normal distribution，但是我們會需要對這個分布做 backpropagation，但是這個分布為不可微分的。因此我們改為從一個固定的分布中聲稱成初始隨機數據 ϵ，並計算 z 作為(ϵ, variance, mean)的函數，也就是 z = mean + variance*ϵ 。

- 實現方法：

  透過對 log variance 取指數後開根號得到 variance，並根據 z = mean + variance*ϵ 計算 z。

- 執行結果：

```
Mean Error 5.639056398351415e-05
Std Error 7.1412955526273885e-06
```

四、 FC-VAE Forward

- 實現方法：

  將輸入送進 encoder 後得到 encoder output，並根據此輸出去計算 mean 跟 log variance。將 mean 和 log variance 用來估計 posterior z (raparametrization)，最後再將 z 送進 decoder 得到還原後的圖片。

五、 Loss Function

VAE 的 loss 如下

$$-E_{Z\sim q_\phi(z|x)}[\log p_\theta(x|z)] + D_{KL}(q_\phi(z|x) \parallel p(z))$$

其中包含 reconstruction loss 和 KL divergence 的部分。

Reconstruction loss 用來計算輸入圖片像素與解碼後圖片像素的 binary cross entropy，KL divergence 是強制潛在空間分布接近先驗分布，這部分需要實現向量化並應用 minibatch。

- 實現方法：

Reconstruction loss 的部分即為 binary cross entropy，可以直接透過 nn.functional.binary_corss_entropy 來實現。KL diverence 可以根據其公式實現，公式為

$$D_{KL}(q_\phi(z|x) \parallel p(z)) = -\frac{1}{2}\sum_{j=1}^{J}(1 + \log(\sigma_{z|x}^2)_j - (\mu_{z|x})_j^2 - (\sigma_{z|x})_j^2)$$

- 驗證結果：

```
Loss error 2.1297676389877955e-06
```

六、 Train a model and Visualize result

利用剛才建立好的 VAE 模型與 loss function 來訓練模型，經過 10 個 epochs 的訓練後，結果如下

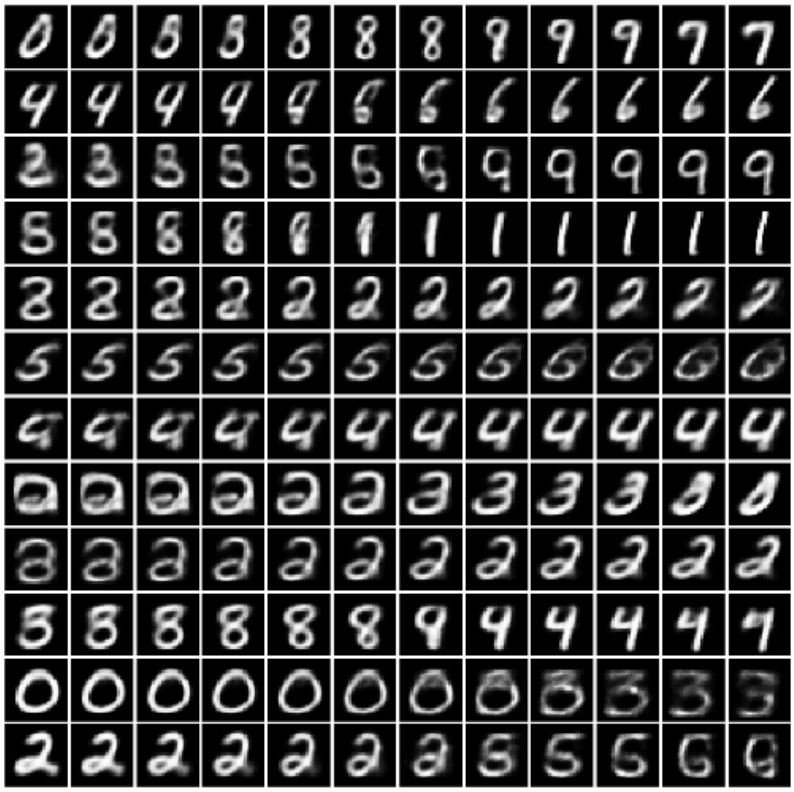
```
Train Epoch: 0  Loss: 157.537354
Train Epoch: 1  Loss: 139.153778
Train Epoch: 2  Loss: 128.003998
Train Epoch: 3  Loss: 124.141708
Train Epoch: 4  Loss: 122.395760
Train Epoch: 5  Loss: 119.549133
Train Epoch: 6  Loss: 115.216003
Train Epoch: 7  Loss: 115.813805
Train Epoch: 8  Loss: 112.929543
Train Epoch: 9  Loss: 108.947662
```

接著我們利用訓練好的 VAE 來生成新的圖片。我們可以初始化某個分布作為 latent space z，並將 latent space 送進 decoder 中來產生新的圖片，產生的結果如下

可以看到大部分結果算是清楚，但是 9、7、6、0 在某些地方會有不容易判別的現象。

七、 Latent Space Interpolation

在 latent space 中，我們可以在兩個 latent vector 之間進行內插，生成新的 latent vector，這些內插後的向量可以用來產生照片，從而觀察模型平滑過度的效果。結果如下



八、 Conditional FC-VAE

Conditional VAE 為 VAE 的變形，他可以利用標籤訊息來進行條件式生成，這樣我們就可以選擇生成特並的類別數據。這邊實作的 CVAE 架構與前面實作的 VAE 架構相同，差別只在於 CVAE 在輸入與 latent space 中加入 one-hot 標籤向量。

- 實現方法：

  使用 VAE 的架構作調整，需要調整個地方如下

  1. Encoder first layer：除了接收 flatten 後的圖像之外，還要去接收一個 one-hot 標籤向量
  2. Decoder first layer : decoder first layer 現在變成是將 latent space + one-hot 向量映射到 hidden dimension。
  3. Forward path：在 forward path 部分，送進 encoder 之前需要將 flatten 後的圖像與 one-hot 向量合併，進到 decoder 之前需要將 latent space 與 one-hot 向量合併。

九、 Train model

用與訓練 VAE 時相同的訓練方式來訓練 CVAE，經過 10 個 epochs 後，訓練結果如下

```
Train Epoch: 0  Loss: 134.811508
Train Epoch: 1  Loss: 126.612228
Train Epoch: 2  Loss: 121.589043
Train Epoch: 3  Loss: 120.414818
Train Epoch: 4  Loss: 117.478844
Train Epoch: 5  Loss: 113.979790
Train Epoch: 6  Loss: 113.741745
Train Epoch: 7  Loss: 114.190109
Train Epoch: 8  Loss: 112.905289
Train Epoch: 9  Loss: 105.665596
```

訓練好模型後，我們可以根據我們想要的類別來產生資料，這邊依序產生 0~9 的數字，結果如下

十、 額外嘗試

- 修改與重新訓練 VAE：

  嘗試修改 VAE 架構以及訓練參數，看看訓練結果會有甚麼影響。

  以下幾種嘗試都是訓練 10 個 epochs。

  1. 將 Linear + ReLU 的層數降到 2：

     調整 encoder 和 decoder 的架構，將 Linear + ReLU 層的數量調整至 2，並訓練 10 個 epochs 結果如下

     ```
     Train Epoch: 0  Loss: 144.756073
     Train Epoch: 1  Loss: 119.712540
     Train Epoch: 2  Loss: 112.382584
     Train Epoch: 3  Loss: 111.661804
     Train Epoch: 4  Loss: 109.088142
     Train Epoch: 5  Loss: 111.145187
     Train Epoch: 6  Loss: 100.483963
     Train Epoch: 7  Loss: 106.553909
     Train Epoch: 8  Loss: 105.120377
     Train Epoch: 9  Loss: 104.061470
     ```

     最終 loss 為 104，利用此模型來生成一些圖片，結果如下

     可以看到圖片跟三層 Linear + ReLU 的模型比起來，更加模糊且更難去判斷數字為多少。

  2. 將 Linear + ReLU 的層數升到 4：

     調整 encoder 和 decoder 的架構，將 Linear + ReLU 層的數量調整

至 4，並訓練 10 個 epochs 結果如下

```
Train Epoch: 0  Loss: 171.944962
Train Epoch: 1  Loss: 147.600037
Train Epoch: 2  Loss: 140.516251
Train Epoch: 3  Loss: 137.701141
Train Epoch: 4  Loss: 138.276703
Train Epoch: 5  Loss: 135.328293
Train Epoch: 6  Loss: 124.415604
Train Epoch: 7  Loss: 126.854019
Train Epoch: 8  Loss: 123.474388
Train Epoch: 9  Loss: 124.066238
```

最終 loss 為 124，比兩層和三層 Linear + ReLU 的模型都還要高。

利用此模型生成圖片，結果如下

由結果可以看到這是目前生成出來最清楚的圖片，每一張都可以很簡單的辨認出數字為多少。

由調整架構的嘗試可以發現到，提高 Linear + ReLU 的層數會造成模型的 loss 提高，但是在生成圖片的表現上，較多 Linear + ReLU 可以生成出更加清楚以及容易辨別的圖片，我猜想這是因為更多層的 Linear + ReLU 可以萃取出更多的特徵，讓還原圖片時有更高的準確度。

3. 提高 latent_size :

架構採用三層的 Linear + ReLU。訓練方式與先前都相同，差別只有將 latent_size 提升到 30。訓練 10 個 epochs 之後結果如下

```
Train Epoch: 0  Loss: 168.234818
Train Epoch: 1  Loss: 138.079895
Train Epoch: 2  Loss: 128.546509
Train Epoch: 3  Loss: 121.915665
Train Epoch: 4  Loss: 119.754166
Train Epoch: 5  Loss: 113.998825
Train Epoch: 6  Loss: 117.927719
Train Epoch: 7  Loss: 121.494675
Train Epoch: 8  Loss: 118.300575
Train Epoch: 9  Loss: 115.604393
```

利用此模型來生成圖片，結果如下

可以看到生成的圖片很多都無法辨別數字，比 latent_size = 15 時的表現還要糟糕。

4. 減少 latent_size :

架構採用三層的 Linear + ReLU。訓練方式與先前都相同，差別只有將 latent_size 降低到 10。訓練 10 個 epochs 之後結果如下

```
Train Epoch: 0  Loss: 158.123764
Train Epoch: 1  Loss: 136.414719
Train Epoch: 2  Loss: 129.768356
Train Epoch: 3  Loss: 120.512810
Train Epoch: 4  Loss: 119.417282
Train Epoch: 5  Loss: 117.790260
Train Epoch: 6  Loss: 119.496498
Train Epoch: 7  Loss: 112.238243
Train Epoch: 8  Loss: 108.600403
Train Epoch: 9  Loss: 107.054176
```

利用此模型來生成圖片，結果如下

可以看到表現也沒有 latent_size = 15 時來的好。

由這次嘗試可以發現到 latent size 並不是越大越好或是越小越好，而是根據使用場景會有合適的大小。要怎麼去決定 latent size，這邊有一些建議 :

- 嘗試不同 latent size，並根據輸出結果選擇合適的 latent size。
- 對訓練數據進行 SVD 奇異值分解，非零的奇異值可以提供一個大概的 latent size 選擇。
- 設定比較大的 latent size 但是在 loss function 中加入 regularization，讓 encoder 被迫只能使用所需要的維度，以減少 overfitting。

- 修改與重新訓練 CVAE

嘗試修改 CVAE 架構以及訓練參數，看看訓練結果會有甚麼影響。

以下幾種嘗試都是訓練 10 個 epochs。

1. 將 Linear + ReLU 的層數降到 2 :

調整 encoder 和 decoder 的架構，將 Linear + ReLU 層的數量調整至 2，並訓練 10 個 epochs 結果如下

```
Train Epoch: 0  Loss: 126.658028
Train Epoch: 1  Loss: 114.523758
Train Epoch: 2  Loss: 109.863144
Train Epoch: 3  Loss: 104.473289
Train Epoch: 4  Loss: 105.329262
Train Epoch: 5  Loss: 101.421585
Train Epoch: 6  Loss: 98.784103
Train Epoch: 7  Loss: 107.049736
Train Epoch: 8  Loss: 95.274269
Train Epoch: 9  Loss: 104.670708
```

最終 loss 為 104，但是在過程中 loss 可以降到 100 以下。利用此模型來生成一些圖片，結果如下

可以看到圖片跟三層 Linear + ReLU 的模型比起來，更加模糊且更難去判斷數字為多少。

2. 將 Linear + ReLU 的層數升到 4：

調整 encoder 和 decoder 的架構，將 Linear + ReLU 層的數量調整至 4，並訓練 10 個 epochs 結果如下

```
Train Epoch: 0  Loss: 148.135284
Train Epoch: 1  Loss: 139.336670
Train Epoch: 2  Loss: 132.594589
Train Epoch: 3  Loss: 134.616745
Train Epoch: 4  Loss: 129.007629
Train Epoch: 5  Loss: 122.334824
Train Epoch: 6  Loss: 128.523071
Train Epoch: 7  Loss: 124.980057
Train Epoch: 8  Loss: 122.108650
Train Epoch: 9  Loss: 112.076660
```

最終 loss 為 112，比兩層和三層 Linear + ReLU 的模型都還要高。利用此模型生成圖片，結果如下

由結果可以看到每一張都可以很簡單的辨認出數字為多少，與先前嘗試過的 CVAE 比較起來也清楚許多。

由調整架構的嘗試可以發現到，CVAE 的結果基本上與 VAE 的結果一樣，這其實是預料之內的，畢竟兩者的架構與運作原理基本上一致。

3. 提高 latent_size :

架構採用三層的 Linear + ReLU。訓練方式與先前都相同，差別只有將 latent_size 提升到 30。訓練 10 個 epochs 之後結果如下

```
Train Epoch: 0  Loss: 142.969391
Train Epoch: 1  Loss: 126.918015
Train Epoch: 2  Loss: 120.830528
Train Epoch: 3  Loss: 112.120186
Train Epoch: 4  Loss: 114.480461
Train Epoch: 5  Loss: 117.759796
Train Epoch: 6  Loss: 115.973068
Train Epoch: 7  Loss: 104.367714
Train Epoch: 8  Loss: 107.558044
Train Epoch: 9  Loss: 108.678383
```

利用此模型來生成圖片，結果如下

可以看到生成的圖片其實算是清楚，跟 latent_size = 15 時的表現相差不大。

4. 減少 latent_size :

架構採用三層的 Linear + ReLU。訓練方式與先前都相同，差別只有將 latent_size 降低到 10。訓練 10 個 epochs 之後結果如下

```
Train Epoch: 0  Loss: 140.689224
Train Epoch: 1  Loss: 119.616302
Train Epoch: 2  Loss: 120.174080
Train Epoch: 3  Loss: 116.448471
Train Epoch: 4  Loss: 116.253418
Train Epoch: 5  Loss: 116.833229
Train Epoch: 6  Loss: 114.153770
Train Epoch: 7  Loss: 110.505920
Train Epoch: 8  Loss: 113.140190
Train Epoch: 9  Loss: 111.449852
```

利用此模型來生成圖片，結果如下

可以看到表現跟之前嘗試的相差不大。

由這次嘗試可以發現到比起 VAE，CVAE 似乎對 latent size 的敏感度沒有那麼高，原因我猜想是因為在訓練 CVAE 時會加入標籤的資訊，模型可以根據標籤去抓取到該數字特徵，有點類似於分類

器的概念，因此在還原或是生成圖片時，模型可以更好的去根據使用者給定的標籤然後根據該標籤之特徵來生成圖片。

十一、 Reference

[1] BC_Wang “變分自編碼器(VAE)的代碼理解”
https://blog.csdn.net/B_C_Wang/article/details/74908408

[2] StackExchange “ What is an appropriate size for a latent space of (variational) autoencoders and how it varies with the features of the images ?”
https://ai.stackexchange.com/questions/37272/what-is-an-appropriate-size-for-a-latent-space-of-variational-autoencoders-and

[5] OpenAI. (2023). ChatGPT (Mar 14 version) [Large language model].
https://chat.openai.com/