Deep Learning (Homework 2)

309555025 資電亥客-羅文笙

1.1 Text Preprocessing

1. 我使用 nltk 套件裡面的 RegexpTokenizer 來使用 regular expression 來切 token 去除掉不必要的符號，regular expression: r'[a-z\-]+。 單純只用空白切 token 的話可能會沒辦法更精確地抓到具有獨立意義的 token，甚至是去除掉一些不必要的 token。

	RegexpTokenizer	**Word_tokenize**	**Split()**
準確度	0.895	0.87	0.85

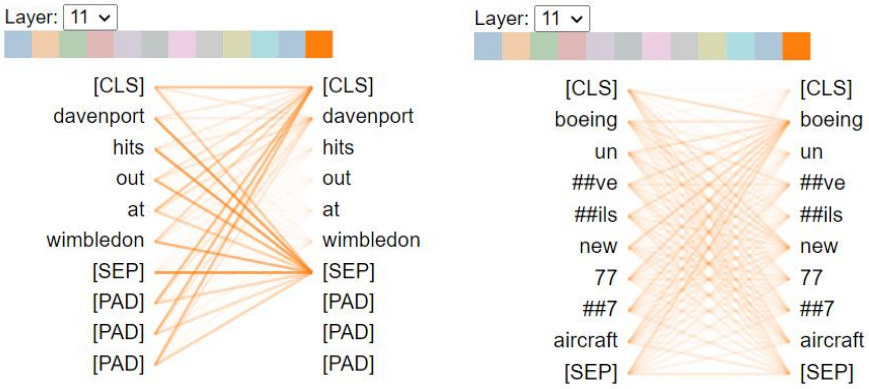
2. <pad>的話主要是為了將每個句子都填成相同長度，以利 batch 來作運算。 <unk>的話是為了描述那些沒有出現在 vocabulary 字典內的字。以上這兩個 special token 都會加強訓練的結果。
3. 我先把 input 都變成小寫，接著用 RegexpTokenizer 去切 token，只留下英文字和有 dash 符號連著的英文字，切著再用 nltk 的 stopwords 去把那些屬於他定義的 stopwords 都刪掉。 在 vocabulary 的建立上，我選擇把 train 和 test 的字都拿來用，並設定 min_count = 1，實驗下來這樣效果最好。最後就是使用 vocabulary 來把每個英文 token 換成數字進去模型裡 train。

1.2 RNN

1. Kaggle 上有準確率有到 0.89
2. 我選擇使用 Glove 的 glove.6B.300d.txt 當作 initial embedding，因為維度越大且資料量越大似乎效果越好，但是太大的 glove 檔，我的記憶體又不夠，只好用這個當作平衡點。使用 pre-trained embedding 能夠讓模型一開始就到一個比較好的起點(Based on 人家 train 好的結果)開始進行學習從而達到全局最佳解，若是使用 random weight 的話比較容易陷入區域最佳解。
3. 我的模型一開始是使 nn.EmbeddingBag，使用這個的話可以讓我不用固定 input 的長度，只要多給一個 offset 即可一次輸入多個 input。接著使用一個 LSTM(input_size = 300, hidden_size=256, layer_num=1)，最後接一個 fully-connected layer(input_size = 256, output = 5)。
   LSTM 的 hidden_size/layer_num 是用實驗的方法測出來的，使用 learning rate=5, batch_size = 32,

1.3 Transformer

1. Kaggle 上的準確率有達到 0.95
2. Show attention map
   (1) 如左圖所示，在 Davenport hits out at Wimbledon 這個新聞標題中，Davenport 和 Wimbledon 得到了最大的 attention，而 Davenport 是一名網球選手，而 Wimbledon 則是知名的溫布頓網球公開賽的意思。模型的預測即為 Sport。
   (2) 如右圖所示，在 Boeing unveils new 777 aircraft 這個新聞標題中 Boeing 和 aircraft 得到的最大的 attention，而 Boeing 是知名飛機公司波音，aircraft 則為飛行器的意思。模型預測即為 Business。

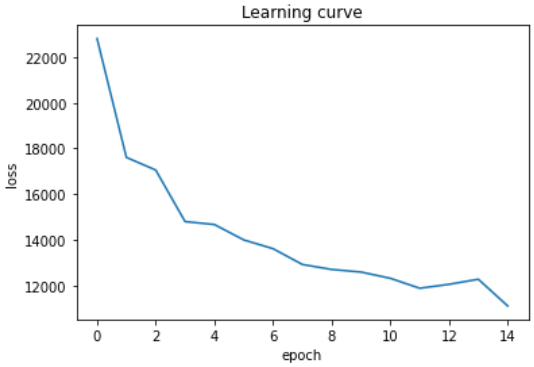


   (3) 以上的結果很合理，Attention 比較大的 token 會對於分類結果有比較大的幫助，所以給他們比較大的權重有助於整體預測正確率的提升。

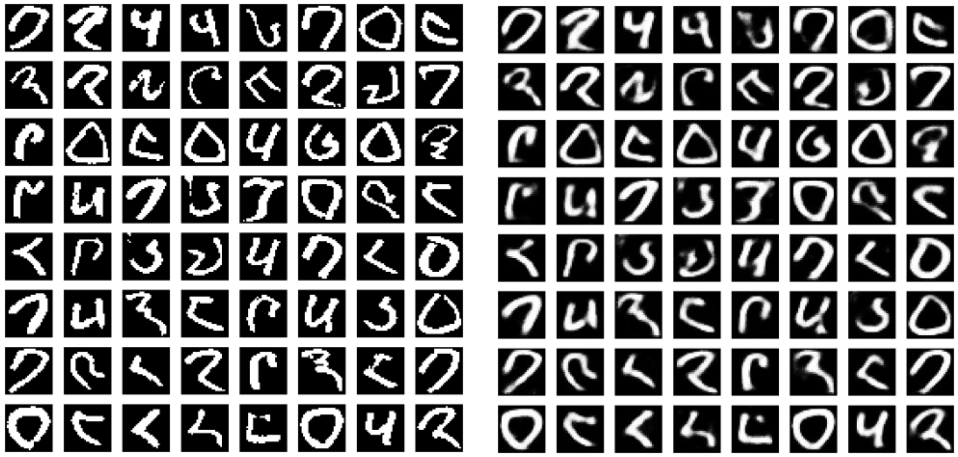
3. 我選擇使用 hugginface 實作的 BertModel，先將 input 的頭尾加上'[CLS]', '[SEP]'，並使用 bert_tokenizer 來切 token，再來使用他 pre-trained 好的 convert_tokens_to_ids 將 token 轉成 id，並加上 padding 使得每個 input 的長度都相同，另外這個 BertModel 還要輸入 masks，所以就是讓 padding 部分的 mask 值設為 0，其他設為 1 即可。
   接下來將 BertModel 的 output 放進 LSTM 去作和 1.2 RNN 部份一樣的操作!
   超參數的設定的話，BertModel 的 input_size=10, hidden_size=784，另外 Learning rate = 1, batch size = 16，LSTM 的 input_size = 784, hidden_size = 128。這些都是透過實驗測試最好的結果。

2. VAE-TibetanMNIST

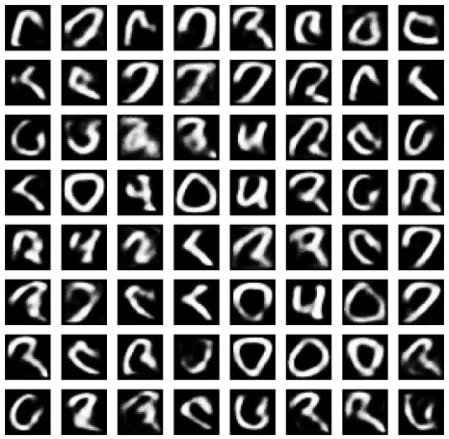
1. Learning curve



(1) some reconstructed samples(左邊為原圖，右邊為重建後的圖)



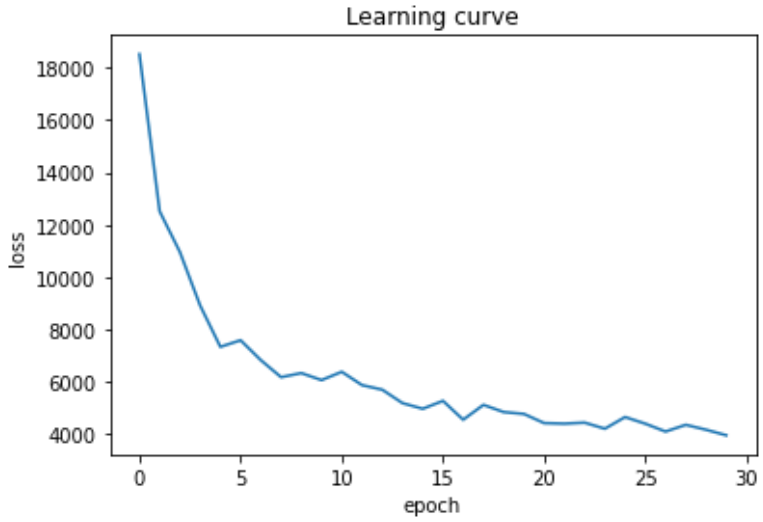
(2) Synthesized image by sample the prior

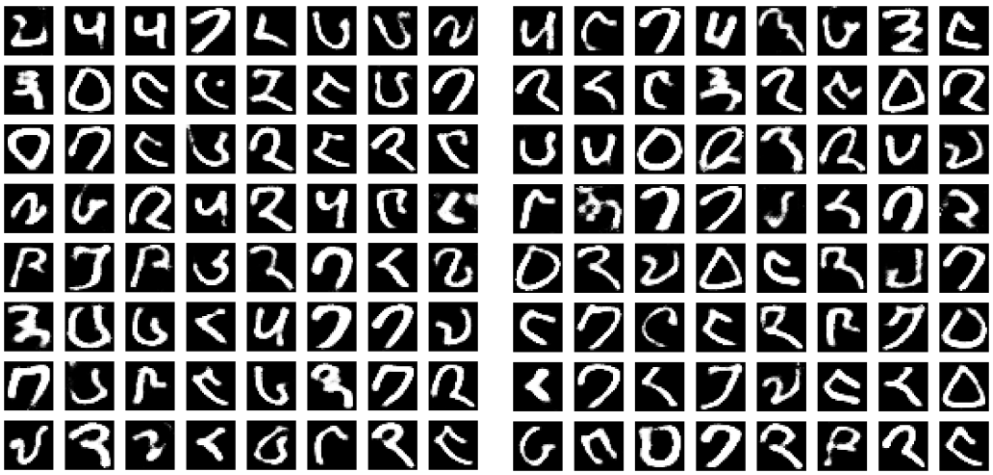


(3) Synthesized Images based on interpolation

(4) Use different λ to show step 1,2,3 ($\lambda = 0$)

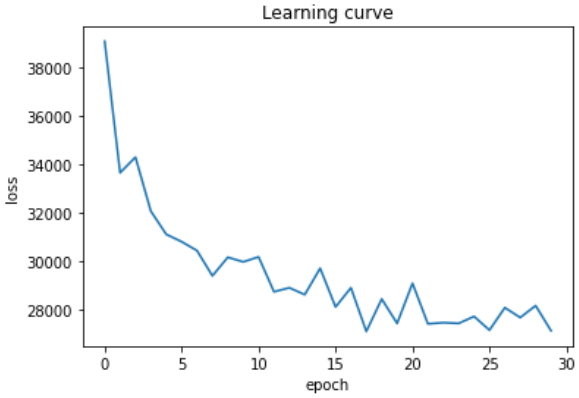
(圖片依序為: Learning curve/reconstruct/ sample from prior /interpolation)

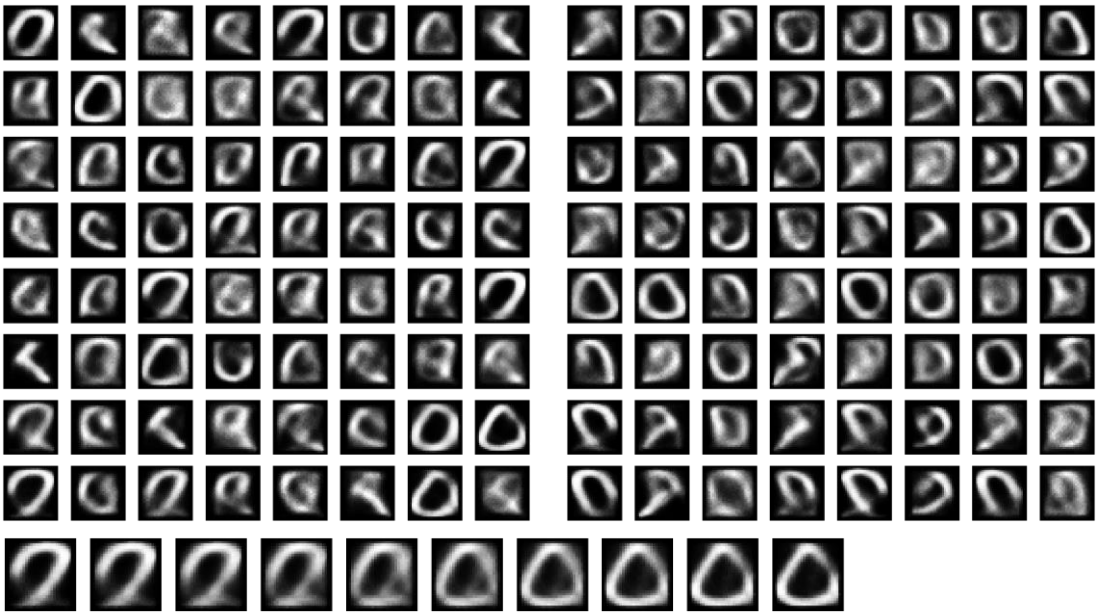




(4) Use different λ to show step 1,2,3 ($\lambda = 10$)

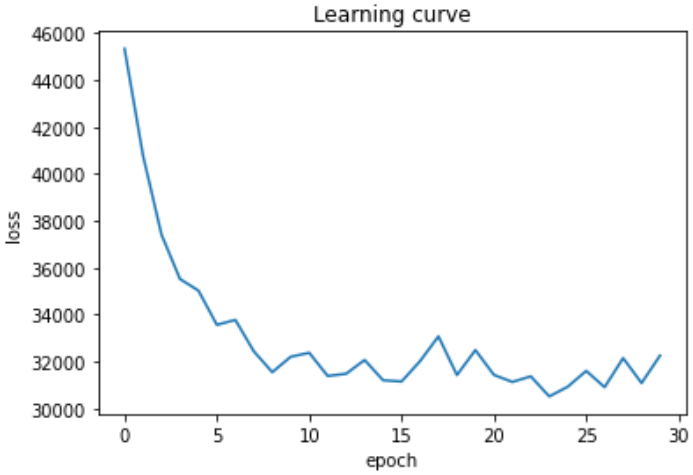
(圖片依序為: Learning curve/reconstruct/ sample from prior /interpolation)

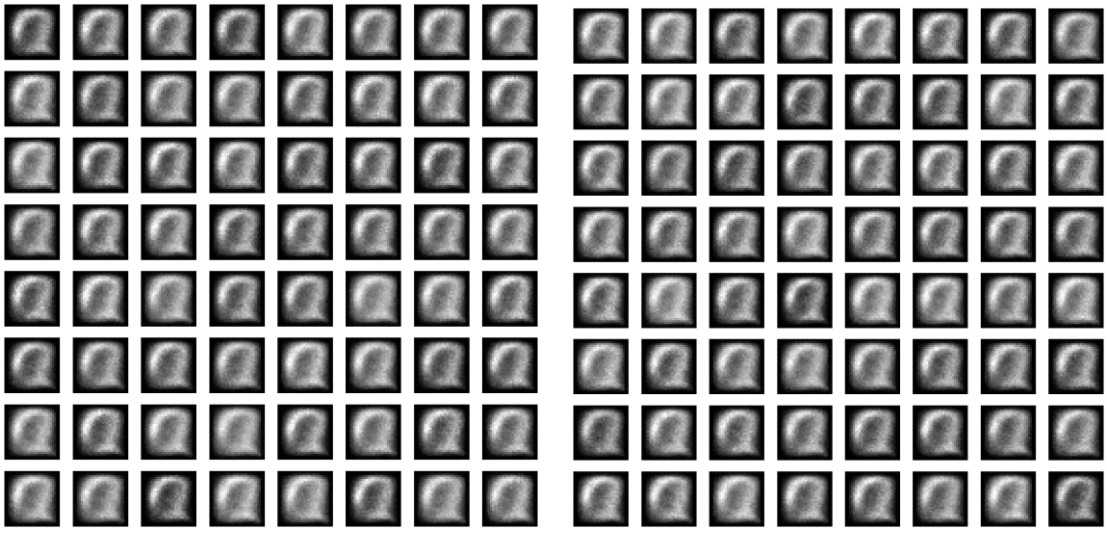


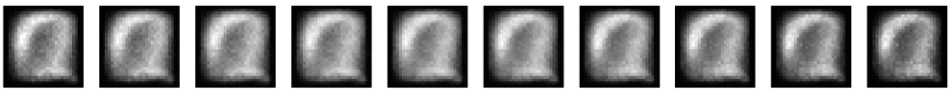


(4) Use different λ to show step 1,2,3 ($\lambda = 100$)

(圖片依序為: Learning curve/reconstruct/sample from prior/interpolation)

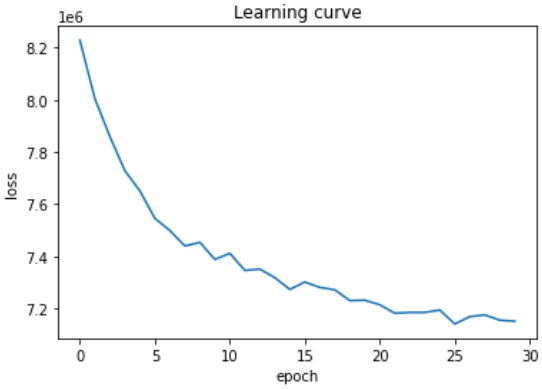






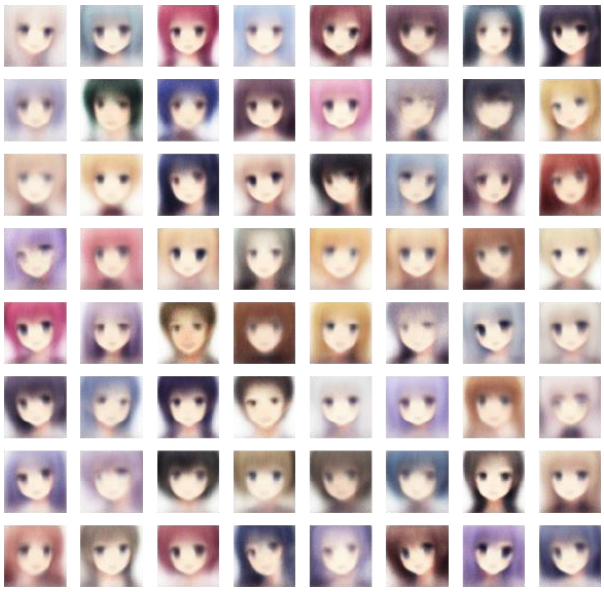
2. VAE-Anime faces

(1) Learning curve



(1) some reconstructed samples(左邊為原圖，右邊為重建後的圖)

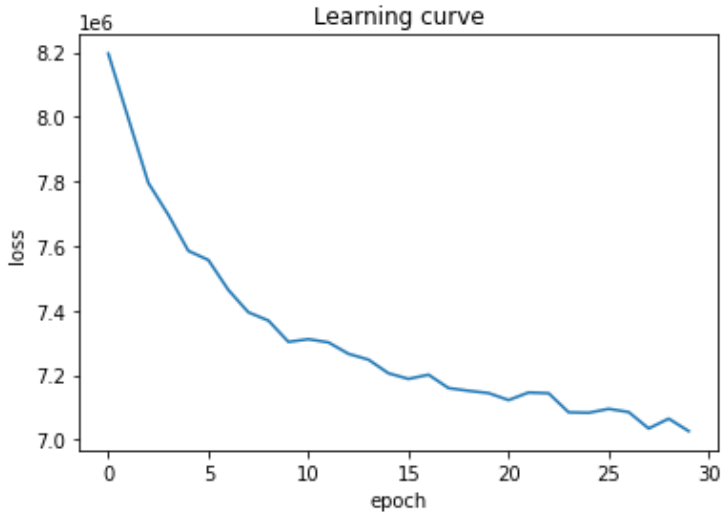
(2) Synthesized image by sample the prior



(3) Synthesized Images based on interpolation

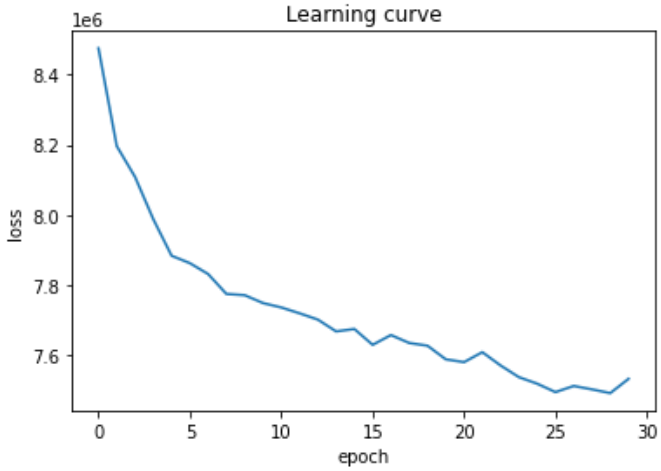
(4) Use different λ to show step 1,2,3 ($\lambda = 0$)

(圖片依序為: Learning curve/reconstruct/ sample from prior /interpolation)



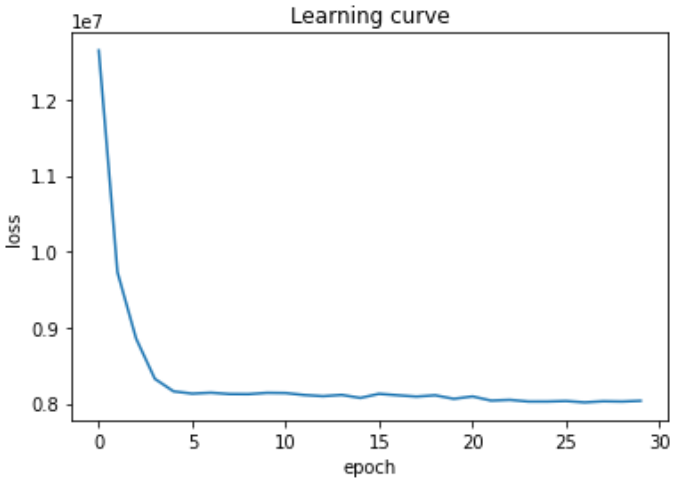
(4) Use different λ to show step 1,2,3 ($\lambda = 10$)

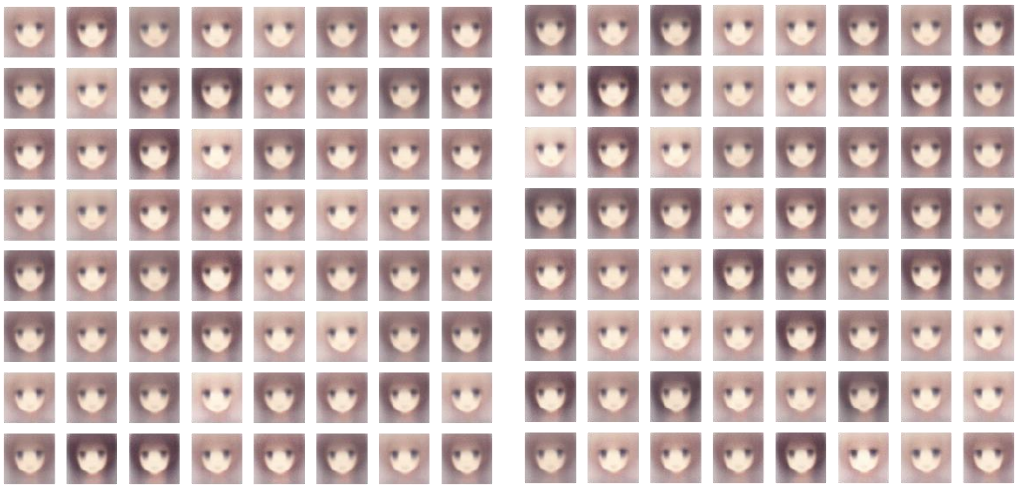
(圖片依序為: Learning curve/reconstruct/ sample from prior /interpolation)



(4) Use different λ to show step 1,2,3 ($\lambda = 100$)

(圖片依序為: Learning curve/reconstruct/ sample from prior /interpolation)





2. VAE-Tibetan MNIST/Anime Faces 模型描述與討論

- MNIST 基本模型超參數:
  (1) λ = 1
  (2)只有一層 hidden layer, size = 400
  (3) Latent vector size = 32
  (4) Batch size = 128
  (5) Learning rate = 0.001
  (6) Epoch = 15
- Anime 基本模型超參數:
  (1) λ = 1
  (2)只有一層 hidden layer, size = 400
  (3) Latent vector size = 32
  (4) Batch size = 2048
  (5) Learning rate = 0.001
  (6) Epoch = 30
- 如何得到 reconstructed samples
  將圖片 input 進入 train 好的模型即可
- 如何從 prior 抽樣 z 來畫出圖形
  儲存模型訓練中產生的 mean, standard deviation，當模型訓練完後再用 Normal distribution 的方式去抽樣產生 z。
- 如何內插兩個 latent codes z 來產生中間的圖片
  使用 Pytorch 提供的函數:torch.nn.functional.interpolate(z, scale_factor=5, mode='linear', align_corners=True) 來將兩個圖片之 latent codes z 之每個 feature 作內插，並將新產生出的這些 z 拿去給 decoder，最後再畫出圖形即可。
- λ 大小之討論:
  λ 的大小可以調整模型一般化的程度，也就是當$\lambda = 0$ 時，整個模型就會盡量的貼近 training set，使得輸入和輸出的圖片非常的相似，但是卻無法在 test set 上有一個比較好的表現。 如果將 λ 調整到一個適當的值讓 Reconstruct loss 和 KL divergence 互相抗衡的話，就會得到一個更好的模型，但是若把 λ 調的太大也會造成不好的效果。