

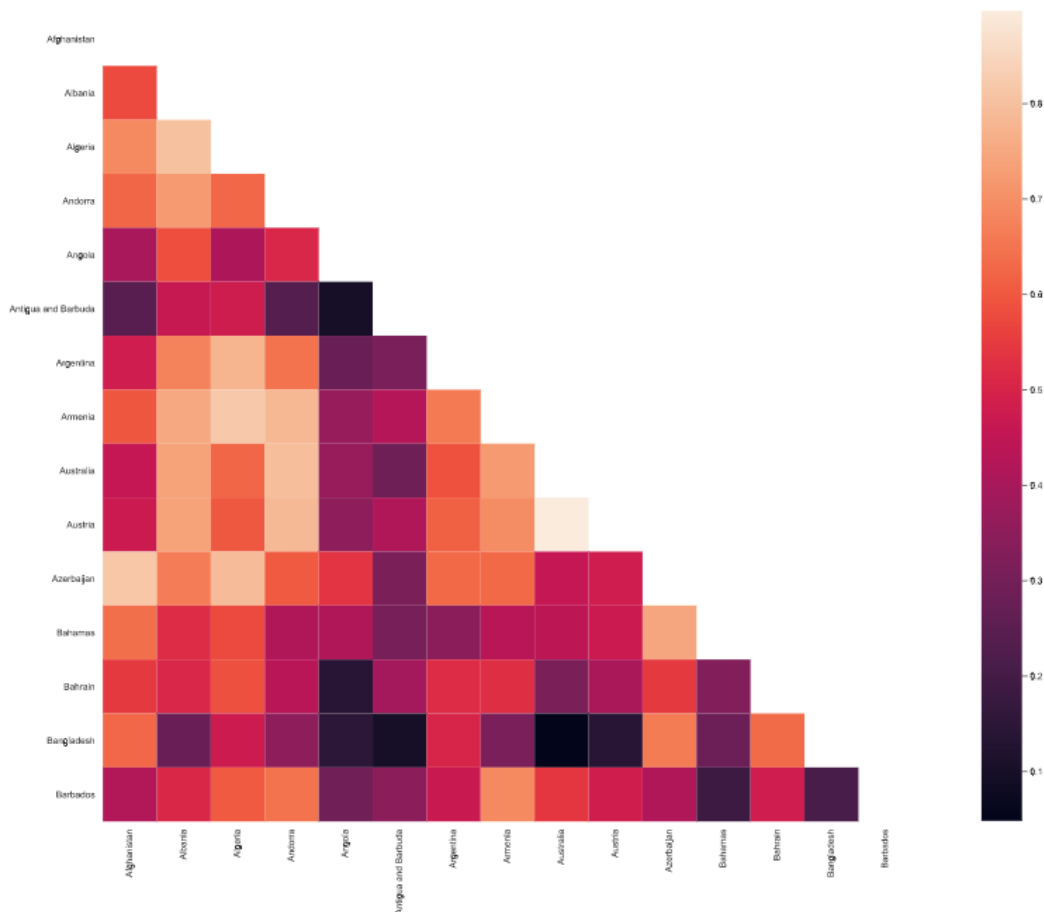
# Deep Learning HW2

## 深度學習作業二

學號：0853412 姓名：吳宛儒

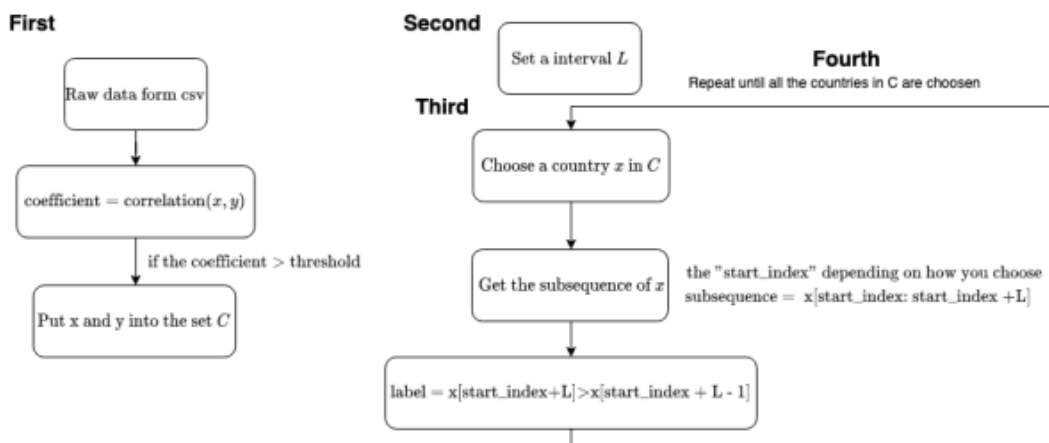
### 1 Recurrent Neural Network for Classification

#### i. Compute correlation coefficient between two countries



因為 185 個國家太多，所以只取了前 15 個國家印出 correlation coefficient。詳細作法在 ii. 中論述。此為對稱矩陣，因此只須印出半部，其他以 mask 遮蔽。對角線上則是皆等於 1。

#### ii. Process on data



主要是遵循助教給的說明文件上的方式。

### **讀取資料與前處理：**

首先因為第 94 行的 Korea, South 有逗號，因此先在 Excel 中手動更改為 Korea South，這樣用 numpy 讀取 csv 才不會有問題。再來先把不需要的欄位以及欄位名稱去除，得到 185 個國家、82 天的資料，也是我們的 data shape。同時也得到了所有的國家名稱(countries)，最後面畫圖會需要用到。

### **計算 coefficient：**

這邊是參照助教在討論區上的說明，因為有 82 天的資料，因此取得 81 個差值，也就是後一天減去前一天得到的差值，所以在 diff\_data 上我們取得了 185 個國家各自的 81 個差值。接著就可以算 coefficient，這裡使用了 numpy 的 corrcoef 函式來做，將國家兩兩計算以後存到 corrcoef\_matrix 裡面。接著便可以透過 seaborn 的 heatmap 畫出 i.的圖。

### **定義 threshold：**

這裡原本取的 threshold 為 0.75~0.8 之間的數值，因為後來加上了後面的處理(取得不重複的資料)，所以決定把 threshold 調低，讓我在這邊可以取到的國家多一點、使得最後不重複的資料可以多一點。因此最後跑實驗的時候是取 0.4。設定這個 threshold 以後，可以得到符合這個條件的 pair 裡面 unique 的國家數總共有 181 個，並將這些國家的 id 存入 C(list)裡面。

### **生成 sequence data：**

這邊一開始的 interval L 設定為 3，後面在 iv.的時候把 L 改設定為 5 觀察差別。作法如同助教文件上載明的，生成序列資料的同時也會生成相對應的 label，如果第三天到第四天是增加的話，label=1，否則 label=0。做完以後會得到 14118 筆序列長度為 3 的資料(seqData)，並得到相對應筆數的 label(seqLabel)數量。

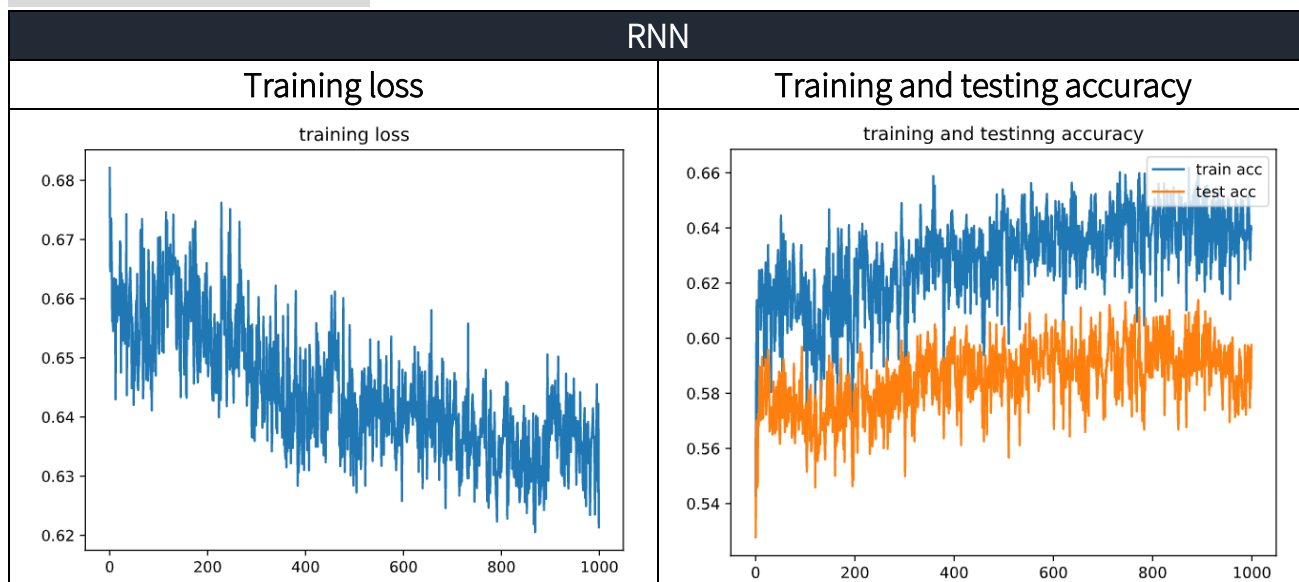
### **取得不重複的資料：**

依據之前寫 DNN 的經驗，如果同樣的 data 對上不同的 label 有可能會讓網路訓練 train 不起來或是容易壞掉，因此在餵進去網路之前，多了此步驟：取得不重複的資料，目的在於確保不會有重複的資料(同 data with 同 label)，以及不會有同 data 不同 label 的資料出現在要放到網路裡面訓練的資料中。首先會先觀察不重複的資料有多長，得到長度為 3688 代表有 3688 個不同的資料，再來從剛剛做好的 seqData 裡面找出不重複的資料(總長度應等於 3688)，以及其對應的 label(選擇第一個遇到的那筆資料的 label 為準)，最後得到總長 3688 的 unique\_seqData 以及 unique\_seqLabel，並統計兩種 label 佔的數量，確保不會有資料不平衡的問題：**1 有 1891 個，0 有 1797 個**，因此判斷目前不需要處理資料不平衡的問題。

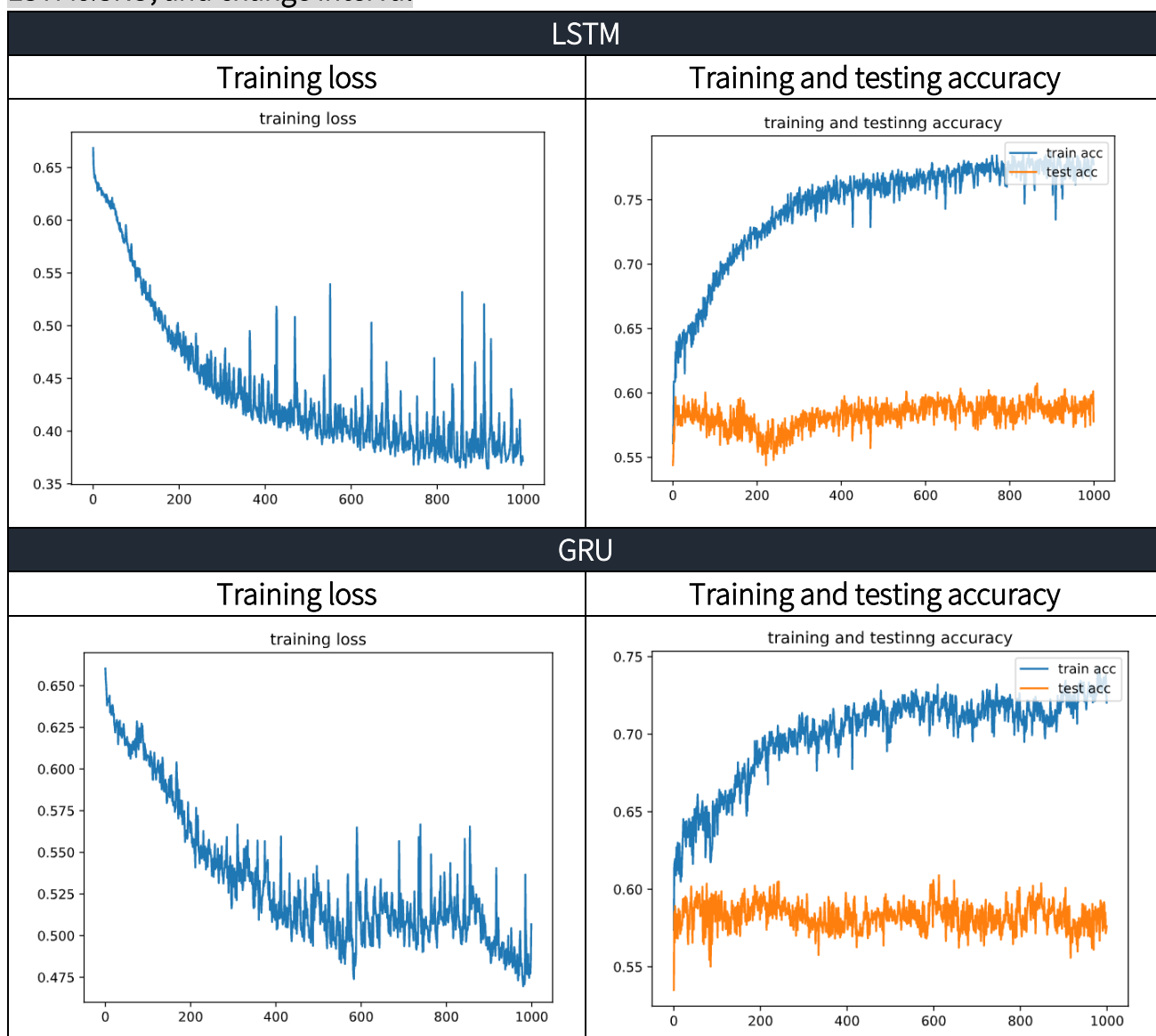
### **網路相關設定**

Learning rate 為 0.009，RNN 或 LSTM 或 gru 皆是出來 64 維度，再經過 linear 出來成一維。實驗初期發覺跑完 RNN 直接出來效果會比較差，且容易 train 不好，因此後面又接了一層。都是只有使用一層來做實驗。最後由 sigmoid 轉回 0~1 之間的數，當作下一天會不會增減的預測。Epoch 數量設定為 1000 次，每訓練 100 個 batch 會印出 train loss 以及 train&test acc。

iii. Recurrent neural network



iv. LSTM&GRU, and change interval



改變 **interval L=5** 以後，再跑了一次 RNN, LSTM, and GRU：

生成 sequence data 以及取得不重複的資料：

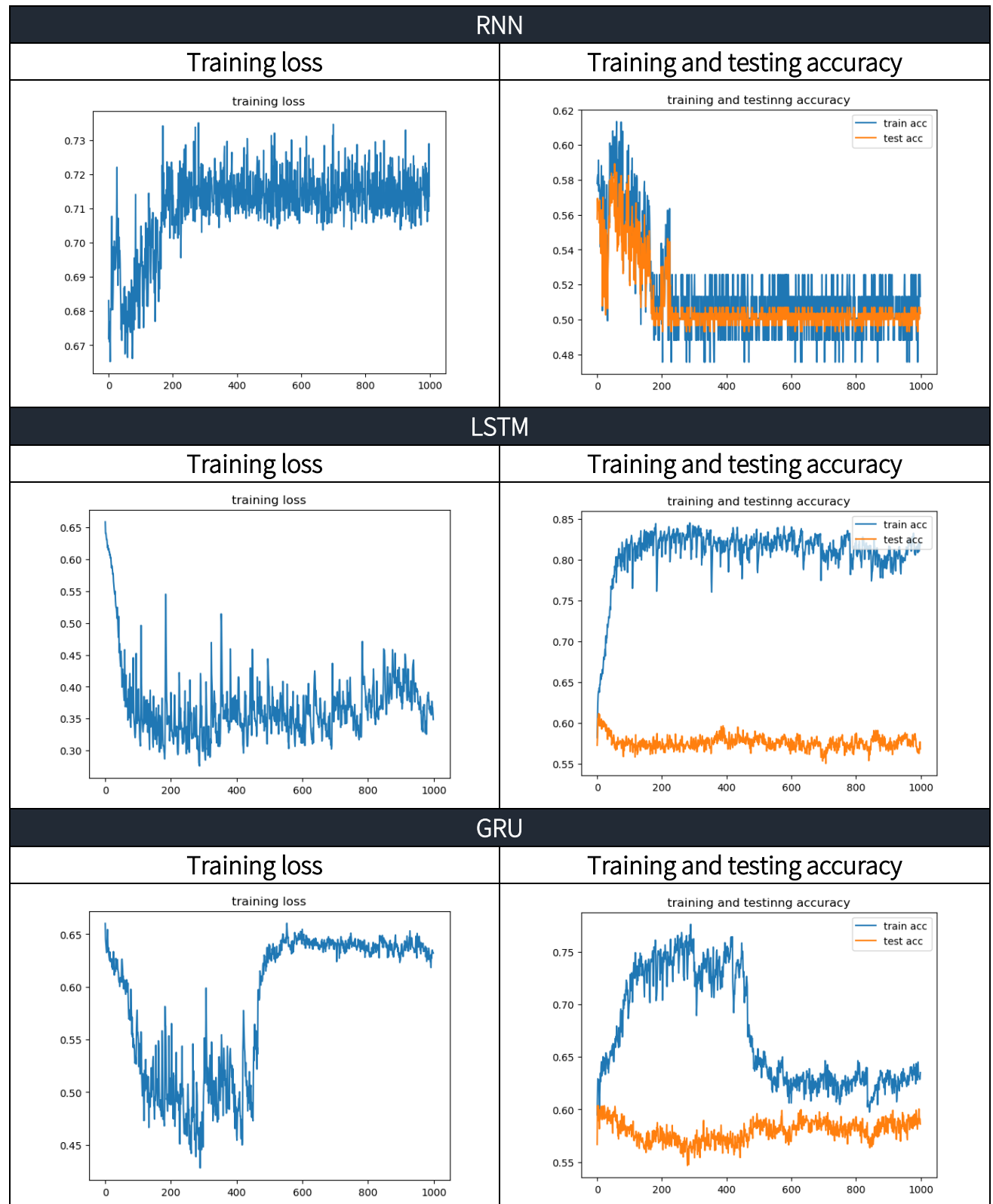
設定 interval L 為 5。做完以後會得到 13756 筆序列長度為 5 的資料(seqData)，並得到相對應筆數的 label(seqLabel)數量。首先會先觀察不重複的資料有多長，得到長度為 4569 代表有 4569 個不同的資料，再來從剛剛做好的 seqData 裡面找出不重複的資料(總長度應等於 4569)，以及其對應的 label(選擇第一個遇到的那筆資料的 label 為準)，最後得到總長 4569 的 unique\_seqData 以及 unique\_seqLabel，並統計兩種 label 佔的數量，確保不會有資料不平衡的問題：1 有 2270 個，0 有 2299 個，因此判斷不需要處理資料不平衡的問題。



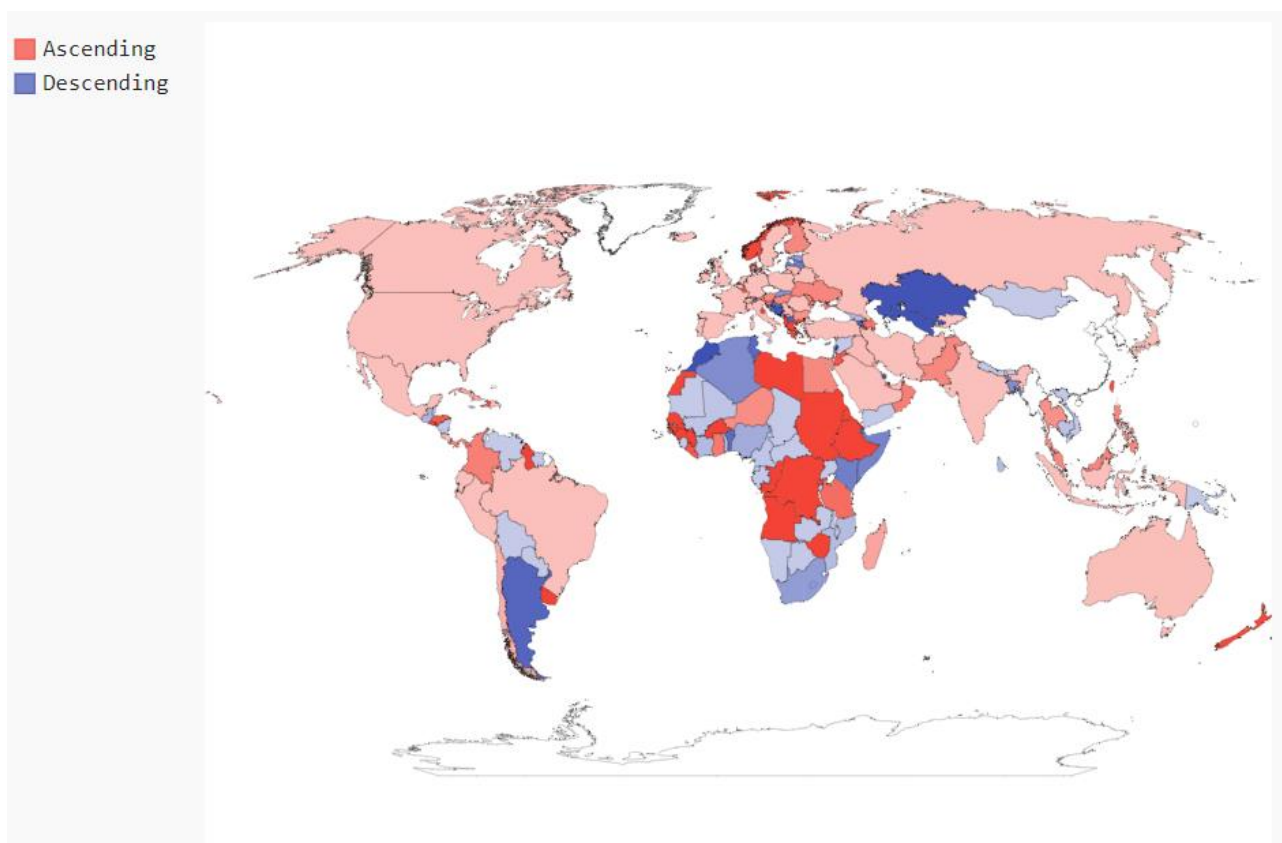
改變 interval L=7 以後，再跑了一次 RNN, LSTM, and GRU：

生成 sequence data 以及取得不重複的資料：

設定 interval L 為 7。做完以後會得到 13394 筆序列長度為 7 的資料(seqData)，並得到相對應筆數的 label(seqLabel)數量。首先會先觀察不重複的資料有多長，得到長度為 5015 代表有 5015 個不同的資料，再來從剛剛做好的 seqData 裡面找出不重複的資料(總長度應等於 5015)，以及其對應的 label(選擇第一個遇到的那筆資料的 label 為準)，最後得到總長 4569 的 unique\_seqData 以及 unique\_seqLabel，並統計兩種 label 佔的數量，確保不會有資料不平衡的問題：1 有 2418 個，0 有 2597 個，因此判斷不需要處理資料不平衡的問題。



## v. Pygal world map plot



L=5 時的 LSTM model 結果。檔案為 world\_map.svg。處理了許多無法對上的國家名稱。在畫圖的時候是把國家名字與預測的值存入兩個 dictionary 裡面，dict\_asc 代表預測值 $\geq 0.5$ 也就是預測會增加的國家，而 dict\_des 則是代表預測值 $< 0.5$ 、預測會下降的國家。

## vi. Discussion

### 關於不同種的 recurrent neural network

從上面的實驗結果可以看出來，RNN 在訓練時不論是 training loss 或是 accuracy 都很震盪且不穩定，這也如同一般所說純粹的 RNN 並不是那麼好訓練。接著是 LSTM 以及 GRU 的部分，這兩者分別用了 RNN 的四倍以及三倍的參數去訓練，可以看到普遍而言，LSTM 效果會比 GRU 來得好，在 computation 允許的運算時間下，雖然 GRU 減少了參數量，但是可能就失去了學習這樣子比較複雜的資料的能力。在資料前處理時，原本若沒有去除重複的資料，訓練的 accuracy 基本上是只有四成、五成的，因此那時便覺得這樣的資料是比較難訓練的，尤其資料裡面出現同 data、不同 label 的情況其實很正常，只是為了讓 model 好一點訓練，所以認為需要先捨棄掉那些冗餘的資料才能夠方便 model 學習。所以，我覺得 LSTM 之所以能表現得比 GRU 來得好，很大的原因在於這次得資料集比較複雜，且只有考慮每天的增減數量，並沒有把國家的其他因素考慮進去，因此很有可能長得一樣的 data 但其實是有隱含其他不同的意義與特徵，但是在我們所擁有的資料上是無法獲得這樣的特徵的(我們只有差值)。當然這樣的有限的資料應該還是可以透過其他方式加入其他特徵，例如在原有的時序列資料以外列入經緯度的考量，讓其成為某個 fully connected layer 上所考量的特徵，一併與原本的 recurrent neural network 一起訓練，或許能達到比較好的效果。

### 關於不同的 interval L (依序使用了 L=3,5,7)

原先使用 L=3 純粹是因為希望一開始不要用太複雜的資料讓 model 不好訓練，但在做了三種不同的 interval 所得到的結果中可以觀察到兩點，第一點是其實 L=7 可以獲得比較多不一樣的資料數量，也就是在前處理後所獲得到的 unique\_seqData 的數量比較多，然而也可能因

為數量較多，同樣的只有一層的 model 無法學習到這麼複雜的資料，因此當  $L=7$  的時候，RNN 和 GRU 訓練到後面都是爆炸的且效果爛掉，loss 飆高、accuracy 急遽降低。GRU 雖然減少了參數量，但可能便因此無法在這樣的資料上有好的效果；第二點為綜合來看， $L=5$  在這樣的網路架構底下，是可以學到比較好的效果的，它比  $L=3$  還能獲得較多的資料數量，比  $L=7$  來得簡單使得 model 能夠學習。

## 2 Variational Autoencoder for Image Generation

### i. Preprocess images and network structure design

#### 資料前處理

作業的 datasets 中，因為所有的 image size 都是(64,64)的彩色圖片，因此是沒有多做 resize 或是 cropping 的，但是在將這些 data 變成 dataset 的時候還是有在 transform 中做了 resize，用來確保所有資料真的都是(64,64)的大小。其他的 transform 則是把圖片轉成 tensor，原本還有做 normalize，但後來發現在後面讀取 batch 的時候資料便已經有正規化過，因此這邊就不再多做 normalize。

#### 訓練集與測試集

接著是 train test 的分割，這邊取 ratio=0.9，也就是 train 和 test 的比例為 9:1，因此處理後得到 train 共 19395 筆，test 共 2156 筆。

#### 參數設定

在網路相關的設定上，選擇使用 batch\_size 為 64，epochs 為 100 次，儲存 sample 的間隔為 100，z 的維度為 32，learning rate 設定為  $1e-3$ 。

#### 網路架構設定

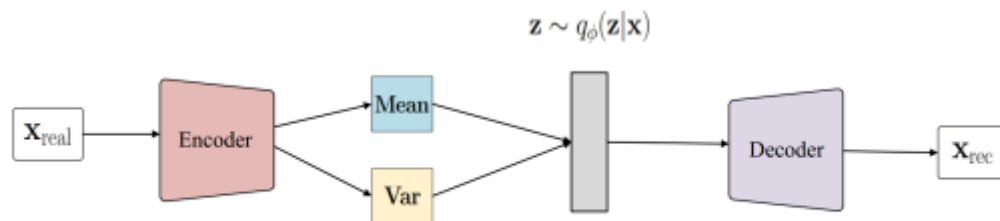


Figure 1: Structure of VAE

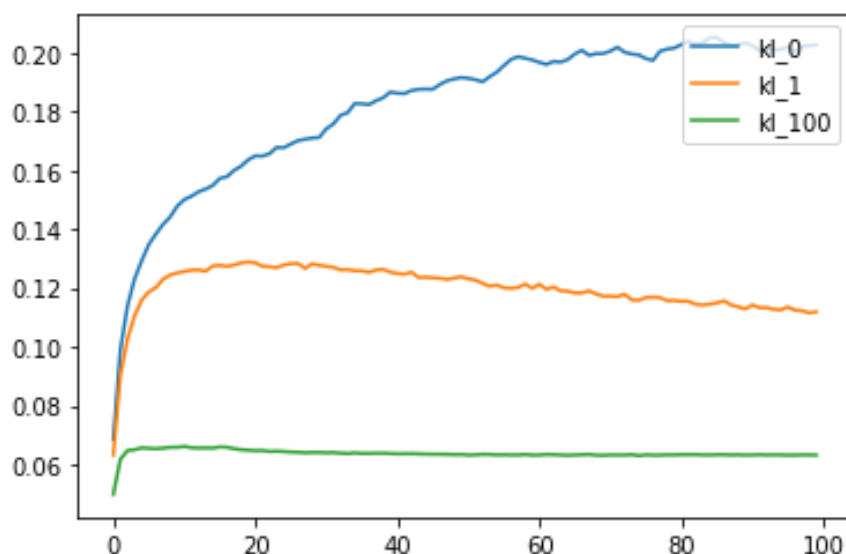
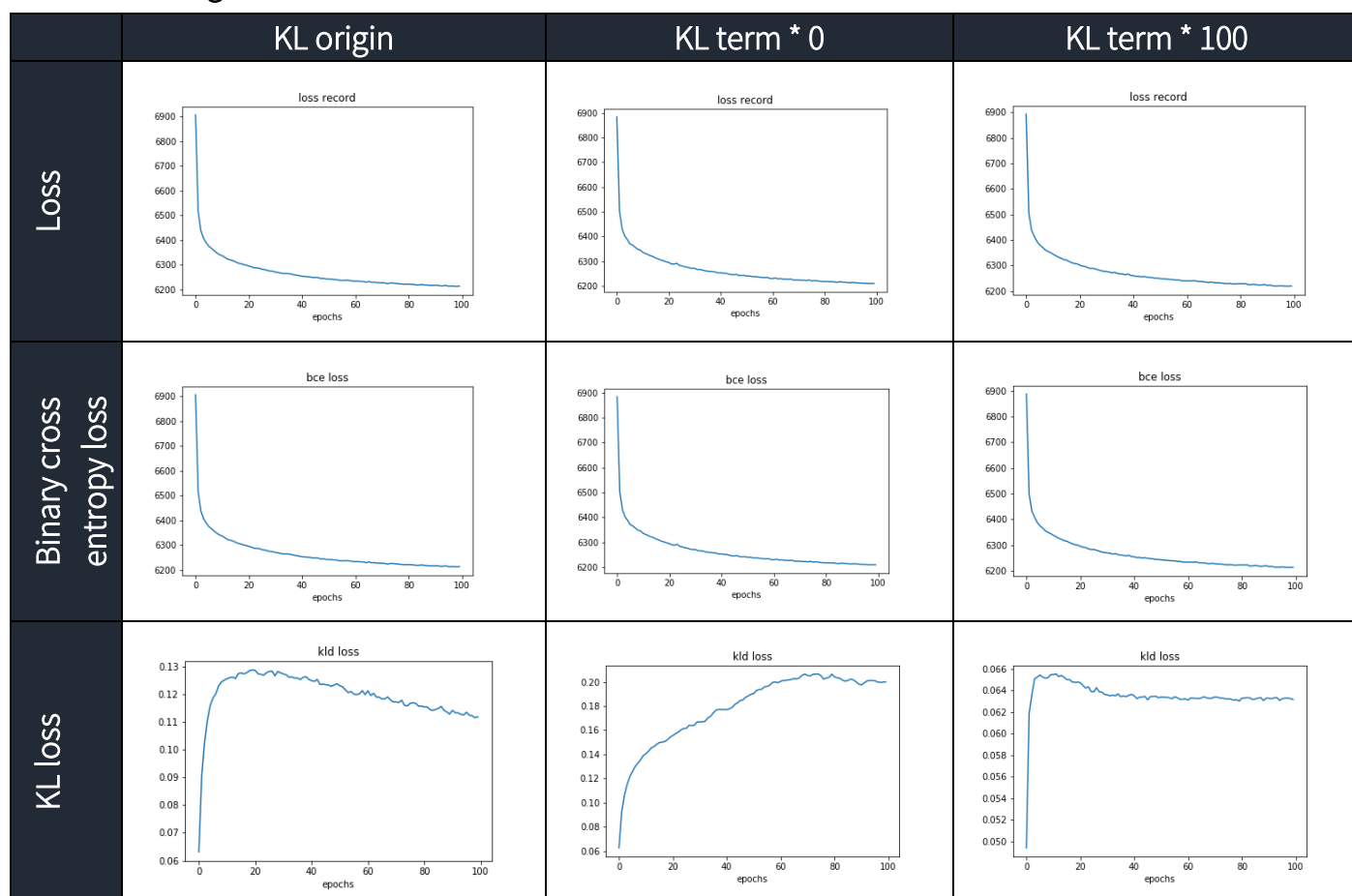
原本最一開始是在 encoder 以及 decoder 中是使用 fully connected 做為主要的網路架構，但是實測以後發現效果並沒有比 CNN 來得好，因此後面的實驗都是使用 CNN 來建構 VAE。最終決定使用了四層的 Conv2D 並在每層之後做 Batch Normalization。而 decoder 則是做 unconv2D。Conv2D 的參數設定在 dims 這個 list 當中，第一個(dims[0])代表圖片是黑白還是彩色，如果是彩色的話則代表有三個通道，所以 dis[0]=3。接著 get\_latent\_z 這個 function 可以取得 encoder 和 decoder 之間的 latent code 以利後續使用，encode 和 decode 兩個 function 則是各自呼叫了 encoder 和 decoder。

#### 圖片儲存設定

在存取上，會在 vae\_results 中有 kl\_0,kl\_1,kl\_100 三個資料夾，各自底下都會有(1)train 資料夾：代表 real 和 reconstructed 的 samples 圖片(for iii.)，還有(2)sample\_z 資料夾：代表 sample  $p(z)$ 在每個 epoch 上透過 z reconstructed 回來以後儲存的圖片(for iv.)，以及(3)interpolation 資料夾：代表透過 z 取差值(size=8)所建構回來的圖片(for v.)。



## ii. Learning curve









從上面的圖可以看出來，三種 KL term 得出來的 loss 以及 BCE loss 是差不多的，雖然在 KL term 上有 0~100 比率的改變，但因為數值很小所以並沒有很顯著地影響著 loss，然而我們也可以很明顯的看出 KL loss 在三種作法上面的明顯差別，畫在同一個圖上的時候如下所示，三種是有很大的差別的。後面會以畫出的圖像結果來討論 KL term 的影響與變化。

## iii. Real samples and reconstructed samples

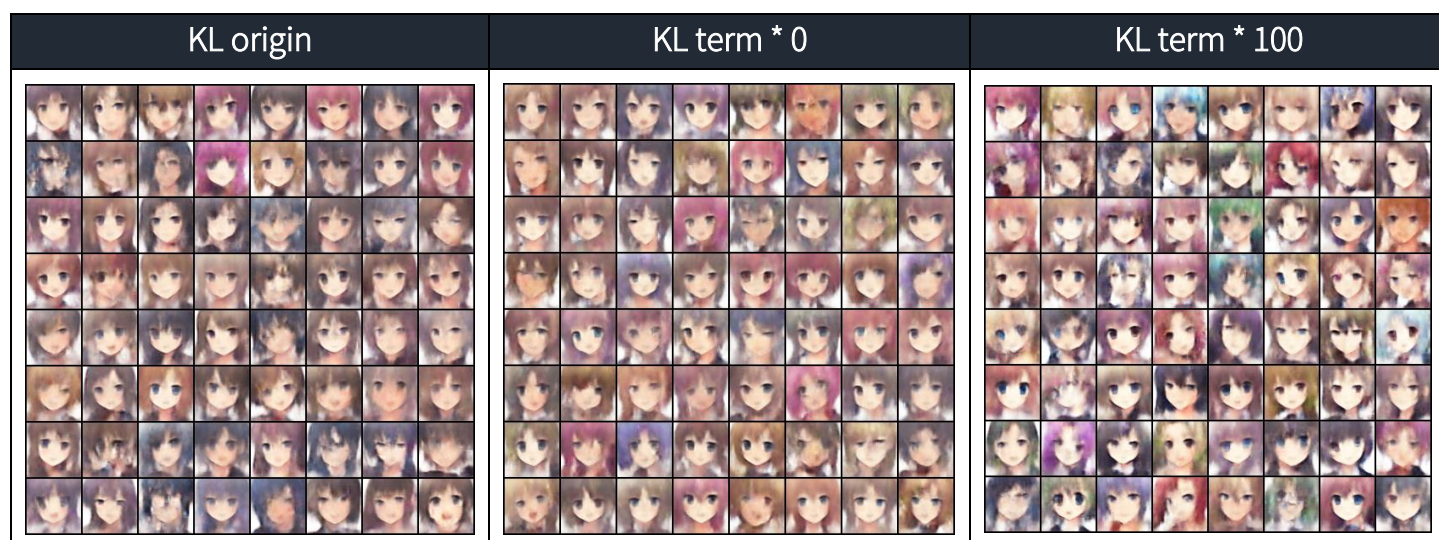
此部分是將原本一個 batch 裡面的原始圖片，和重構回來的圖片相互比較。普遍而言，我認為三者的效果是差不多的，而 KL\*100 的色彩的鮮豔度比較高一些，不會像 KL\*0 的時候會有點糊糊的。在實驗過程中也遇到 vae 重構回來的圖都是灰灰的，雖然看的到人形但就像套了灰色的濾鏡，後來才發現是多加了一層 sigmoid，拿掉以後就正常許多。



	Real samples	Reconstructed samples
KL origin		
KL term * 0		
KL term * 100		



iv. Sample  $p(z)$  and synthesized samples



此部分則是先在訓練模型之前先 sample 一個(batch\_size, 32)大小的隨機值做為 sample  $p(z)$ 。將三張圖一起比較，可以明顯看出 KL\*100 的圖片中明顯比其他有比較多樣的髮色，也有比較鮮艷的結果，而在 KL origin(也就是 KL\*1)和 KL\*0 上則沒有太大的差別，前者稍微比後者好一些。也有和其他使用不同網路架構的同學討論，發現如果是在 encoder decoder 都是使用 fully connected 的架構，在 KL\*0 模型重構出來的圖片會壞掉，也就是沒有生成「人的樣子」，而在 KL 大的時候會比較 general，這是和我的結果比較不一樣的部分，估計是不同的網路會生成不一樣的結果，而在 CNN 的架構底下，KL\*0 時並不會壞掉，KL\*100 的時候也可以比 KL origin 有較好的結果。

v. Synthesizes images based on the interpolation of 2 latent codes  $z$  between 2 real samples

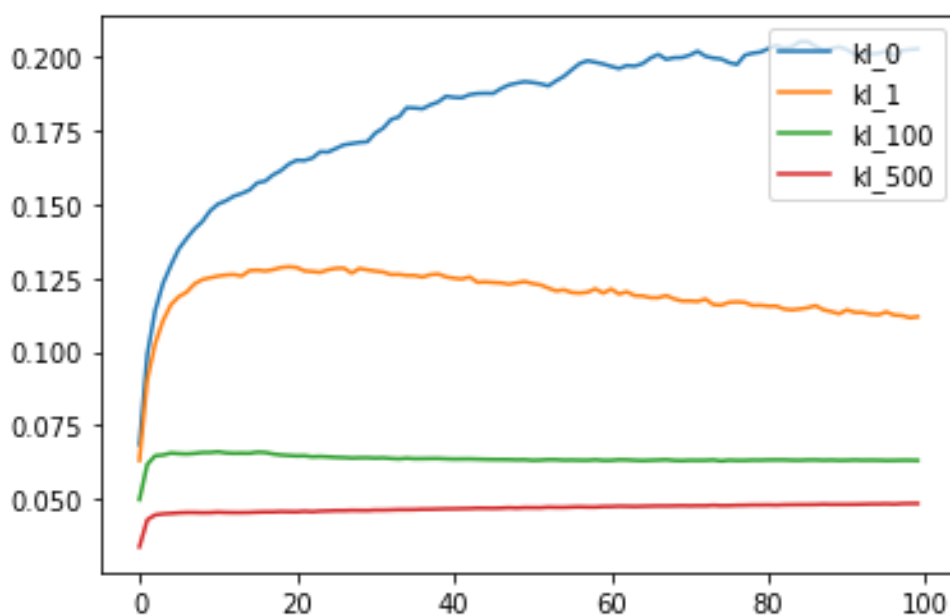




在插值的部分，是先取兩張原始圖片經過 vae 的 encoder 以後透過 `get_latent_z` 這個 function 取得所謂的 latent code，再將這兩個 latent code 中間取插值，直到最後包含原始兩條 latent code，總共有八條 latent code，再將這八條 latent code 餵到 decoder 去生成 reconstructed 的圖片。結果如上所示。普遍來看我覺得如果取出來的兩張圖片並沒有很顯著的差別，那其插值的效果並沒有差太多。不過還是可以大概看出  $KL \times 100$  的時候其髮色從左到右是有顯著變化的，雖然在臉上面都是同個表情(沒有訓練好)，但髮色的轉變已經比上面兩個明顯許多，而  $KL$  origin 雖然也有顯著的髮色差別(第二行)，但其實和原始的兩張圖片的髮色相去甚遠，因此還是判定在重構上面還是沒有訓練的很好，另外  $KL \times 0$  的部分我認為其髮色幾乎完全不在原本的兩張圖片上或根本只有一種髮色、只有一種樣子，因此認為這是三者之中效果最差的。

#### vi. Discussion on KL term based on result

如同上面 ii.~v.所討論的結果，發現在我所設定的網路架構底下， $KL \times 100$  或許能得到比較好的結果，而  $KL$  本身就是在計算兩個分布的相似程度，如果完全相同， $KL$  會等於零，其餘皆會是大餘零的數，因此套用在實驗結果上，當 loss 中的  $KL$  被重視、強調時( $\times 100$ )，model 的確在原始圖片以及重構的圖片上面更加地把兩者的分布拉近，另一個角度來看，其實重構回來的圖都會比較模糊也比較灰濛濛的，然而  $KL \times 100$  重構回來的圖片相較於其他兩者就比較鮮艷，也比較接近原始圖片的樣子，因此會判定這樣的強調  $KL$  loss 的手法在這樣的架構上是有顯著效果的。因為好奇  $KL$  可以大到甚麼程度，因此後來又做了  $KL \times 500$  的實驗，其  $KL$  loss 與前三次的實驗曲線如下：



而也試著將剛才所討論的圖片都一併印出來觀察：

這是將原始圖片與重構圖片對比的結果，經過 100 次 epoch 以後，可以看到  $KL \times 500$  所重構出來的圖片雖然還是一樣鮮豔，但好像有越來越 general 的趨勢，例如眼睛，中間部分所生成的圖片很多的眼睛都差不多，不像剛才的  $KL \times 100$  有比較多樣化的眼睛。





接著左邊是 sample  $p(z)$  以後所生成的結果，可以看到如同前段所述，眼睛幾乎都長得一樣。甚至出現了不規則形狀。

而在插值的部分，則是看出了比較不同的部分。不同於前三次實驗，雖然這次效果仍沒有很顯著，但是可以看出來其插值所生成出來的圖片之髮色有些微差別，但如果髮色沒有明顯不同的兩個原始圖片做插值的話，生成出來的圖片幾乎長得完全一樣。



但其中也找到了效果不錯的例子，在頭髮長度上有明顯的插值：由長而短。



整體而言，我還是認為除了  $KL*500$  有偏向 general 的趨勢，但還是和  $KL*100$  一樣是比其他兩種方式能得到比較好的結果的。