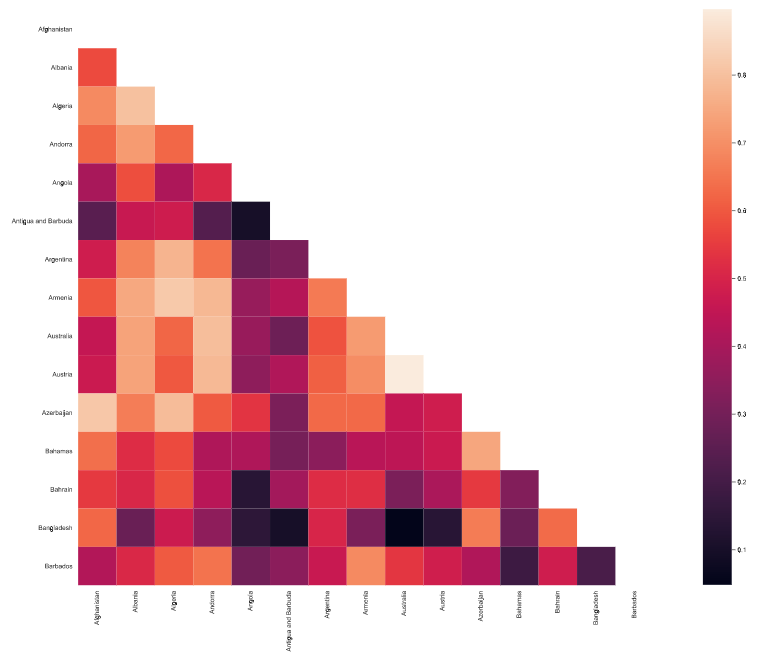
**Deep Learning HW2**

**深度學習作業二**

學號：0853412 姓名：吳宛儒

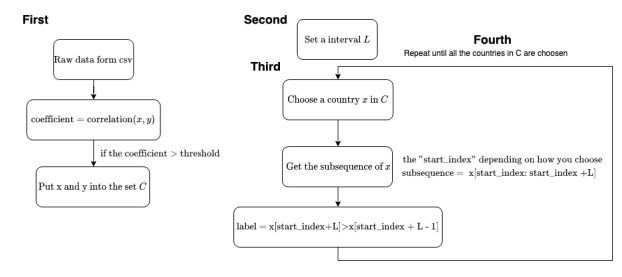
**1 Recurrent Neural Network for Classification**

1. **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。

1. **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。

1. **Recurrent neural network**

|  |  |
| --- | --- |
| **RNN** | |
| **Training loss** | **Training and testing accuracy** |
|  |  |

1. **LSTM&GRU, and change interval**

|  |  |
| --- | --- |
| **LSTM** | |
| **Training loss** | **Training and testing accuracy** |
|  |  |
| **GRU** | |
| **Training loss** | **Training and testing accuracy** |
|  |  |

**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。

|  |  |
| --- | --- |
| **RNN** | |
| **Training loss** | **Training and testing accuracy** |
|  |  |
| **LSTM** | |
| **Training loss** | **Training and testing accuracy** |
|  |  |
| **GRU** | |
| **Training loss** | **Training and testing accuracy** |
|  |  |

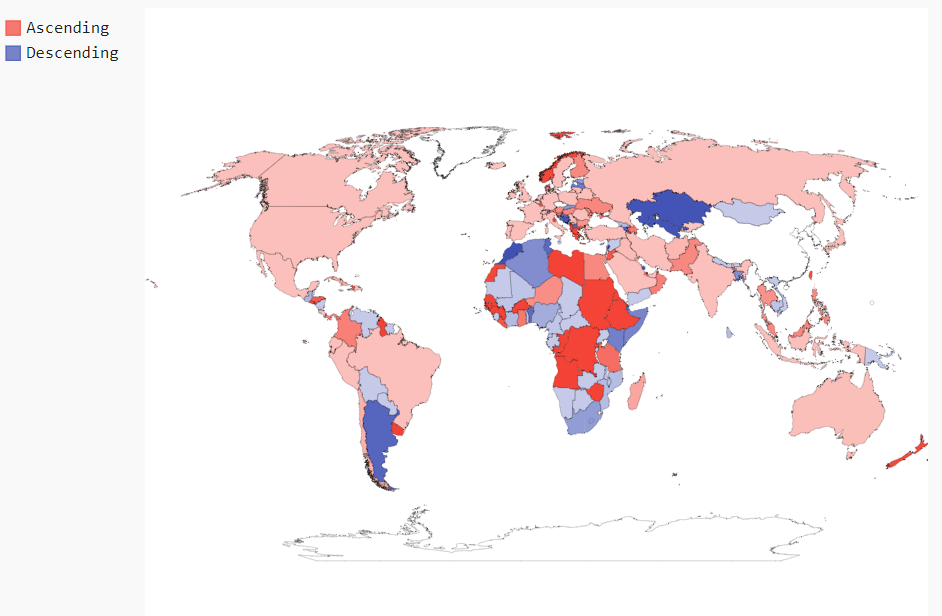
**改變interval L= 7以後，再跑了一次RNN, LSTM, and GRU：**

**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。

|  |  |
| --- | --- |
| **RNN** | |
| **Training loss** | **Training and testing accuracy** |
|  |  |
| **LSTM** | |
| **Training loss** | **Training and testing accuracy** |
| C:\Users\wwj\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\E41875EA.tmp | C:\Users\wwj\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\2FCF5588.tmp |
| **GRU** | |
| **Training loss** | **Training and testing accuracy** |
|  |  |

1. **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>=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。

1. **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**

1. **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。

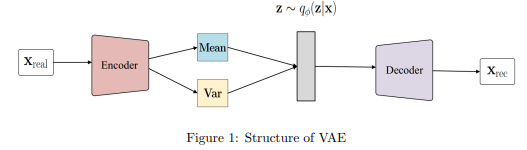
**訓練集與測試集**

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**

****

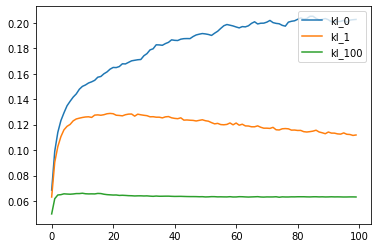
原本最一開始是在encoder以及decoder中是使用fully connected做為主要的網路架構，但是實測以後發現效果並沒有比CNN來得好，因此後面的實驗都是使用CNN來建構VAE。最終決定使用了四層的Conv2D並在每層之後做Batch Normalization。而decoder則是做unconv2D。Conv2D的參數設定在dims這個list當中，第一個(dims[0])代表圖片是黑白還是彩色，如果是彩色的話則代表有三個通道，所以dis[0]=3。接著get\_latent\_z這個function可以取得encoder和decoder之間的latent code以利後續使用，encode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

1. **Learning curve**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **KL origin** | **KL term \* 0** | **KL term \* 100** |
| **Loss** |  |  |  |
| **Binary cross entropy loss** |  |  |  |
| **KL loss** |  |  |  |

****

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。

1. **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** |  |  |

1. **Sample p(z) and synthesized samples**

|  |  |  |
| --- | --- | --- |
| **KL origin** | **KL term \* 0** | **KL term \* 100** |
|  |  |  |

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。

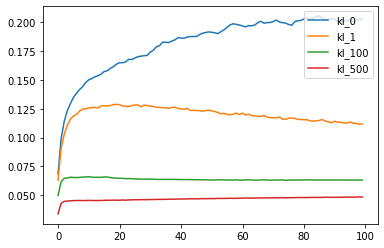
1. **Synthesizes images based on the interpolation of 2 latent codes z between 2 real samples**

|  |  |  |
| --- | --- | --- |
|  | **Original image** | **Interpolation** |
| **KL origin** |  |  |
| **KL \* 0** |  |  |
| **KL \* 100** |  |  |

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\*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\*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。

1. **Discussion on KL term based on result**

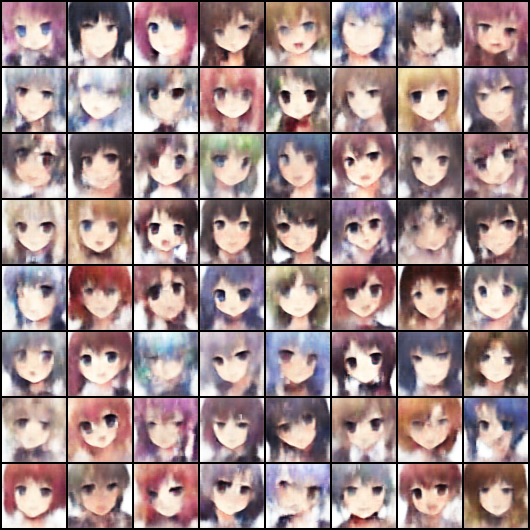
如同上面ii.~v.所討論的結果，發現在我所設定的網路架構底下，KL\*100或許能得到比較好的結果，而KL本身就是在計算兩個分布的相似程度，如果完全相同，KL會等於零，其餘皆會是大餘零的數，因此套用在實驗結果上，當loss中的KL被重視、強調時(\*100)，model的確在原始圖片以及重構的圖片上面更加地把兩者的分布拉近，另一個角度來看，其實重構回來的圖都會比較模糊也比較灰濛濛的，然而KL\*100重構回來的圖片相較於其他兩者就比較鮮艷，也比較接近原始圖片的樣子，因此會判定這樣的強調KL loss的手法在這樣的架構上是有顯著效果的。因為好奇KL可以大到甚麼程度，因此後來又做了KL\*500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