

## 1. Recurrent Neural Network for Classification :

### ■ Network Architecture and Performance with Confusion Matrix:

#### ◆ Parameters :

Batch_size	20
learning_rate	0.0001
L	7

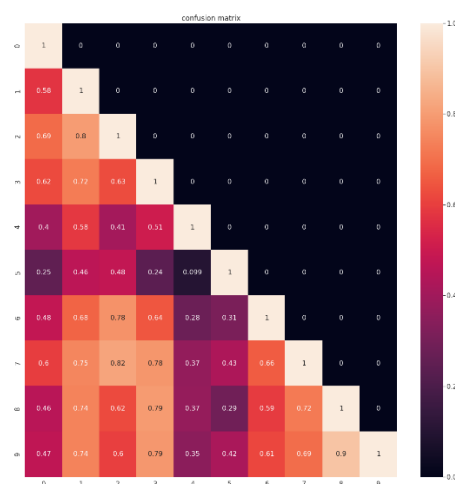
#### ◆ Step of Experiment :

##### ● Step 1 :

將蒐集的資料包成矩陣，再用 `numpy` 的 `diff` 進行運算，`numpy` 的 `diff` 是將下一個 `column` 減去上一個 `column` 的值，因此藉由 `numpy diff` 算出來的矩陣的物理意義可視為「增加趨勢」的矩陣，利用此矩陣進行訓練才有趨勢的 `pattern` 可以學。

##### ● Step 2:

將 `step 1` 求出來的矩陣用 `numpy` 的 `corrcoef` 算出 `correlation` 所形成的矩陣，矩陣求出後得到如下所示的圖形。



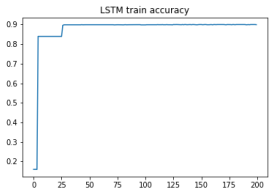
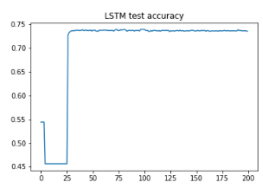
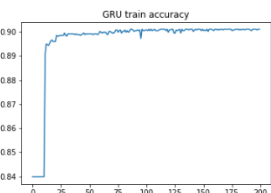
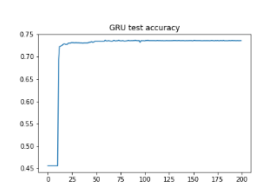
##### ● Step 3 :

利用 `correlation` 所形成的矩陣進行國家的挑選，我用一個 `loop` 對此矩陣的每一個 `row` 去 `loop` 完一輪，在 `loop` 每一個 `row` 時挑出大於 0.55 的元素，將這些元素集成一個 `list`，若此 `row` 的元素個數大於一個 `threshold`(ex.在此中我設 40)，代表這個國家與其他 40 個國家有正相關，且相關係數會大於 0.55，所以就將這個國家選起來放進去我待會要用於 `train` 的 `list` 內。

##### ● Step4 :



將這些選起來的國家以 `L` 切割成片段放入學習的模型內。

◆ Accuracy performance :

	Train	Test
LSTM		
GRU		

在此之中模型可以很快的學到是否有增加的 Pattern，但是結果其實出乎意料的有點讓人沮喪，如下結果所示。(我試了蠻多方法，整理如下)

◆ Result :

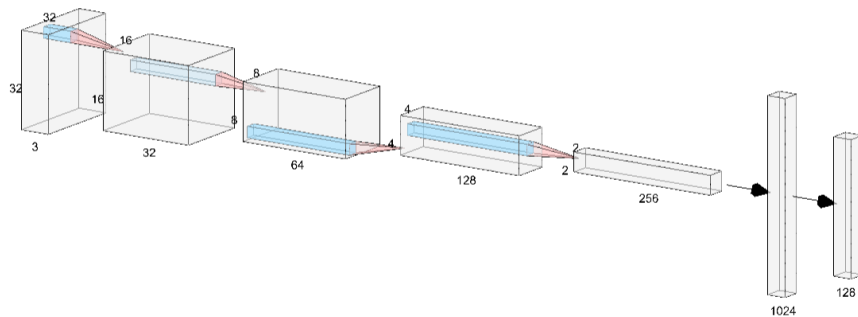
LSTM	
GRU	

- 因為 csv 讀進來的國家事後看了一下，其實準確度不太高，我利用 pygal 提供的 COUNTRIES 以名稱的相符程度做出對應的簡稱 list 並放入用於指示 list 中，在該 list 中其實錯誤率很高。
- 正確率的 pattern 其實正確率上升得太快而且上升之後幾乎水平不動，我檢查過 training data 應是正確的，模型也沒有問題，嘗試改掉 time step 就 train 不動了 QQ。

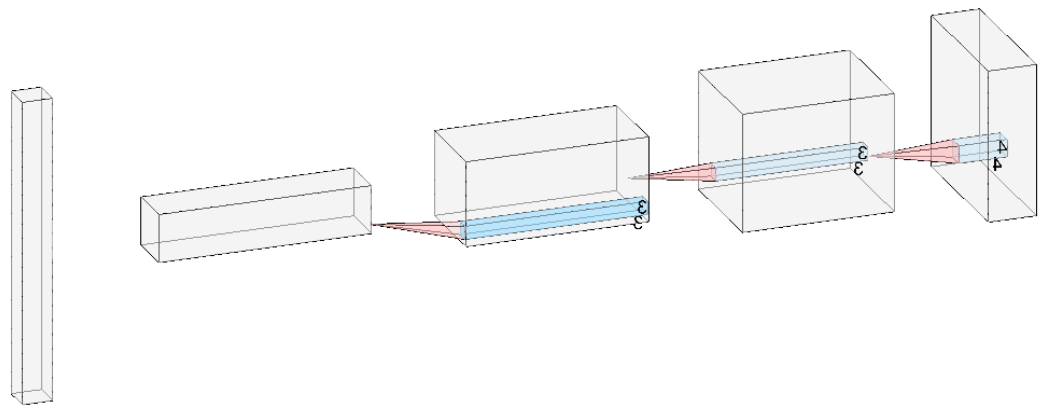
## 2. Variational Autoencode for Image Generation :

### ■ Network Architecture:

#### ◆ Encoder



#### ◆ Decoder :



### ■ Preprocessing Method :

- ◆ Step-1 : 先利用 PIL 將 image 轉換成 RGB 格式
- ◆ Step-2 : 為了統一訓練資料間的平衡，將資料 reshape 成模型 input 的大小，並將資料整理成一個 numpy 的 matrix
- ◆ Step-3 : 圖形集成一個 matrix 後，再利用一個 random list 進行 matrix 的 shuffle。
- ◆ Step-4 : 將 data normalize 後再餵入 model 內進行訓練。

### ■ Loss function design :

使用 MSE Loss 與 KL Divergence，KL Divergence 使用[1]所提到的，其物理意義是使 encoder 的 latent 經過 decoder 後的 output 與原來資料的分布接近的方法。KL Divergence 就是在降低此兩分佈的距離間的量測指標。

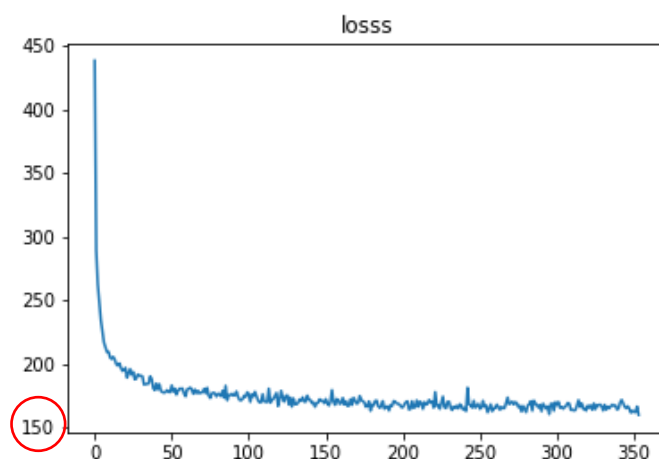
■ Experiment Result & Analysis : (使用 test.py 繪製)

◆  $KL * 1 + MSE * 1$  :

■ Patameter :

Batch_size	30
learning_rate	0.001
Training_data	10000

■ Loss performance :



對於此 Model 來說 Loss 的 Lower bound 大約在 150 附近。抓在此附近的 Loss 經驗上來看表現較其他的好，所以可以將 Epoch 拉長與 Loss function 上進行設計，以達到此 Loss。

■ Graph Result :

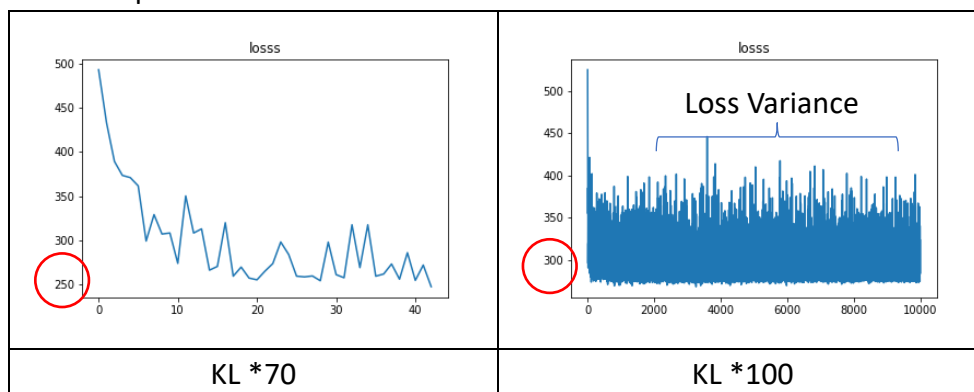
圖 形	Example and Synthesis	Example	Synthesis
	Interpolated		
Analysis		<ol style="list-style-type: none"> <li>1. Synthesis 的結果可明顯改變髮色與人物的輪廓。</li> <li>2. 基於 Synthesis 的特性，利用將 latent variable 進行線性轉換，將圖形轉換成其他風格。在 Interpolated 的部分我使用兩張圖做內插。兩張圖人物的髮色差異極大，分別為深咖啡與淺粉色，利用內插值的連續分布特性，可將兩圖所得到的圖形之間用連續分布的方式展現出漸漸變化的效果，這是常見的風格轉換應用的方法之一。</li> </ol>	

◆  $KL * 70 + MSE * 1$  與  $KL * 100 + MSE * 1$  :

■ Patameter :



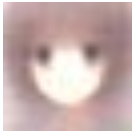
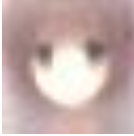


Batch_size	50
learning_rate	0.0001
Training_data	12000

■ Loss performance :



隨著 KL Divergence 的占比越高，Loss 的 Lower bound 也越來越高，代表模型的優化能力較差，因為降低一小點的 Loss 在 Loss function 所形成的平面上移動的速度較快，所以容易產生在訓練時 Loss 的 Variance 越來越高的現象。

■ Graph Result :

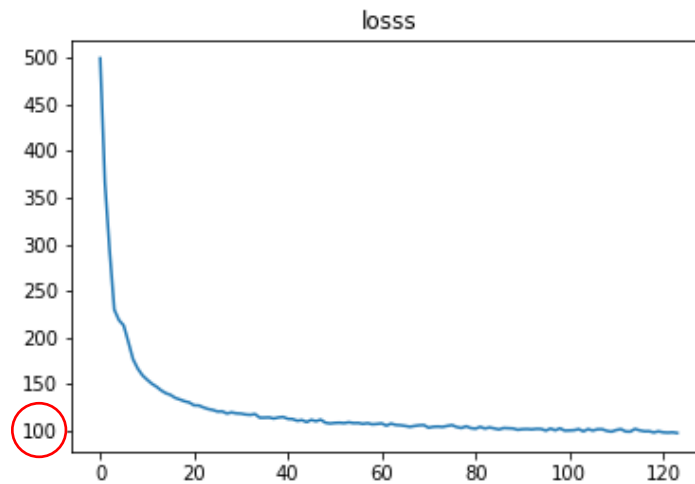
		Example	Synthesis
Example and Synthesis	KL * 70		
	KL * 100		
Interpolated	KL * 70		
	KL * 100		
Analysis		<ul style="list-style-type: none"> <li>● Loss 越高可使匯出的圖形較為 Smooth，但是對 Latent 的線性轉換變化的敏感度降低，導致相同的 Synthesis 中，髮色與肌膚變化較小。</li> <li>● 在 Interpolated 的部分使用兩張圖做內插。因為對 Latent Variable 的線性變化反應的靈敏度隨著 KL 佔比大越來越下降，所以做 Interpolated 變化越來越不明顯。</li> </ul>	

◆  $KL * 0 + MSE * 1$  :

■ Patameter :

Batch_size	50
learning_rate	0.0001
Training_data	12000

■ Loss performance :



隨著 KL Divergence 的佔比下降，Loss 的 Lower bound 越來越下降，遠比前面的三個 Case 都還要來的小，代表 loss function 所形成的平面較緩，所以訓練過程中的 variance 也非常的小。

■ Graph Result :

圖 形	Example and Synthesis	Example	Synthesis
	Interpolated		
Analysis		<ul style="list-style-type: none"> <li>● Synthesis 的結果相較於前面的例子可明顯改變髮色與人物的輪廓，但是圖形的解析度遠比前面的 case 還要來的差。</li> <li>● 基於 Synthesis 的特性，利用將 latent variable 進行線性轉換，但是此處遺憾的點是因為解析度下降太多，而使得實驗看起來沒有前面的好，推測原因是因為 Loss 佔比下降，所以 docode 的分佈與原圖形分佈的差異較大，造成 decoder 的能力太差，而使得圖形的解析度大大下降。</li> </ul>	

Reference :

[1] Kingma and Welling. Auto-Encoding Variational Bayes. ICLR, 2014