

DLHLP HW2 Voice Conversion Report

組長 github id: ming024

組員：r08922080 資工碩一 簡仲明

r08921062 電機碩一 黃健祐

b04501127 土木四 凌于凱

HW2-1 (Auto-Encoder) (2.5%)

- (1) 請以 Auto-Encoder 之方法實做 Voice conversion。如果同學不想重新刻一個 auto-encoder，可以試著利用[這個repo](#)的部分程式碼，達到實現出 auto-encoder。如果你是修改助教提供的 repo，請在 report 當中敘述你是如何更改原本程式碼，建議可以附上修改部分的截圖以利助教批閱；同時，如果各位有更動原本模型參數也請一併列出。如果你的 auto-encoder 是自己刻的，那也請你簡單敘述你的實作方法，並附上對應程式碼的截圖。(1%)

./hps/vctk.json：把train encoder跟decoder的step改成100000，其他改成0

```
21 |         "enc_pretrain_iters": 100000,  
22 |         "dis_pretrain_iters": 0,  
23 |         "patch_iters": 0,  
24 |         "iters": 0
```

./hps/en_speaker_used.txt

```
1 | 1  
2 | 2
```

./main.py：把不是train encoder跟decoder的地方註解掉（39、40、41行）

```
38 |         solver.train(args.output_model_path, args.flag, mode='pretrain_G')  
39 |         #solver.train(args.output_model_path, args.flag, mode='pretrain_D')  
40 |         #solver.train(args.output_model_path, args.flag, mode='train')  
41 |         #solver.train(args.output_model_path, args.flag, mode='patchGAN')
```

./solver.py：新增178、179兩行

```
175 |         if iteration % 100 == 0:  
176 |             for tag, value in info.items():  
177 |                 self.logger.scalar_summary(tag, value, iteration + 1)  
178 |             if iteration % 1000 == 0 or iteration + 1 == hps.iters:  
179 |                 self.save_model(model_path, iteration)
```

- (2) 在訓練完成後，試著將助教要求轉換的音檔轉成 source speaker 和 target speaker 的 interpolation，也就是在 testing 的時候，除了將指定的音檔轉成 p1 和 p2 的聲音之外，請嘗試轉成 p1 和 p2 interpolation 的聲音。並比較分析 interpolated 的聲音和 p1 以及 p2 的關係。你可以從聲音頻率的高低、口音、語調等面向進行觀察。只要有合理分析助教就會給分。請同時將這題的音檔放在 github 的 hw2-1 資料夾中，檔名格式請參考投影片。(1.5%)

做法：

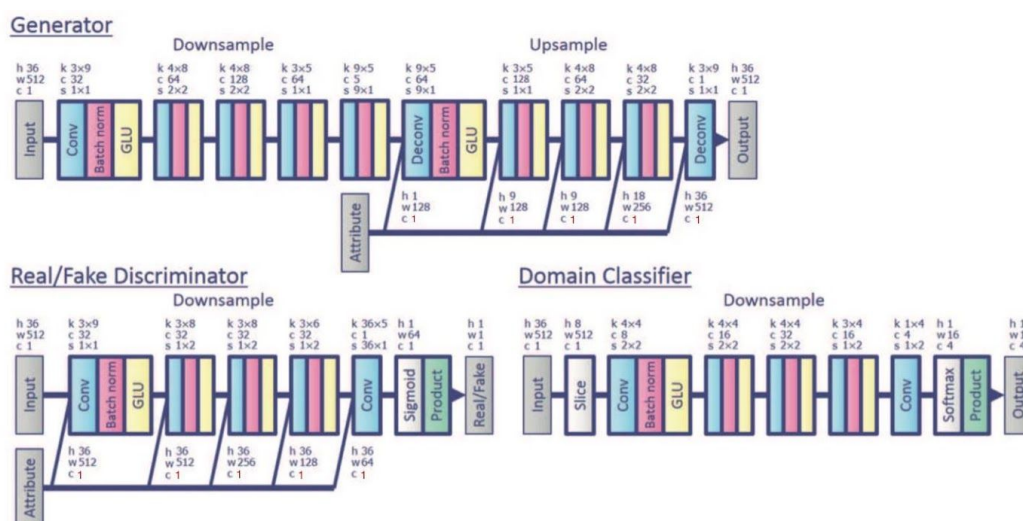
將decoder裡面的五個emb layer個別用[0]以及[1]當input，將兩emb output取平均，再個別放回原本的layer input。

分析：

字詞前面p1_p2_inter比較接近p1的聲音，而p2_p1_inter則比較接近p2的聲音，兩種字詞後面都含有對方的音調，但成份較小，猜測可能是content encoder還有保留一些speaker information，導致與embedding的平均合起來，source的聲音比較明顯，而兩者中間值的聲音在斷詞原本聲音成份較小的地方才顯示出來。

HW2-2 (GAN) (2.5%)

- (1) 請使用助教在投影片中提到的連結，進行 voice conversion。請描述在這個程式碼中，語者資訊是如何被嵌入模型中的？請問這樣的方式有什麼優缺點？有沒有其他的作法可以將 speaker information 放入generator 裡呢？(1%)



將語者資訊c(one hot encoding) 在feature與time的方向repeat成與input相同，再把它與input 延channel dimension的方向串接起來。

這樣作法實做起來容易，但參數量與語者數量成正比，語者越多，則model參數量會變多，因為每個convolution layer的filter要多出語者數量的維度。

可以將c通過speaker embedding layer，然後將hidden layer拿出來跟其做inner product，再與conv的input concat，這樣參數量就固定，不會隨語者數量而變化。

- (2) 請描述你如何將原本的程式碼改成訓練兩個語者的 voice conversion 程式。

(0.5%)

因為speaker information變成[0]或[1]，而不是大小為4的one hot encoding，因此需要修改model會用到speaker information的地方。

./model.py

```

65 | self.up1 = Up2d(6, 64, (9,5), (9,1), (0,2))
66 | self.up2 = Up2d(65, 128, (3,5), (1,1), (1,2))
67 | self.up3 = Up2d(129, 64, (4,8), (2,2), (1,3))
68 | self.up4 = Up2d(65, 32, (4,8), (2,2), (1,3))
69 |
70 | self.deconv = nn.ConvTranspose2d(33, 1, (3,9), (1,1), (1,4))

```

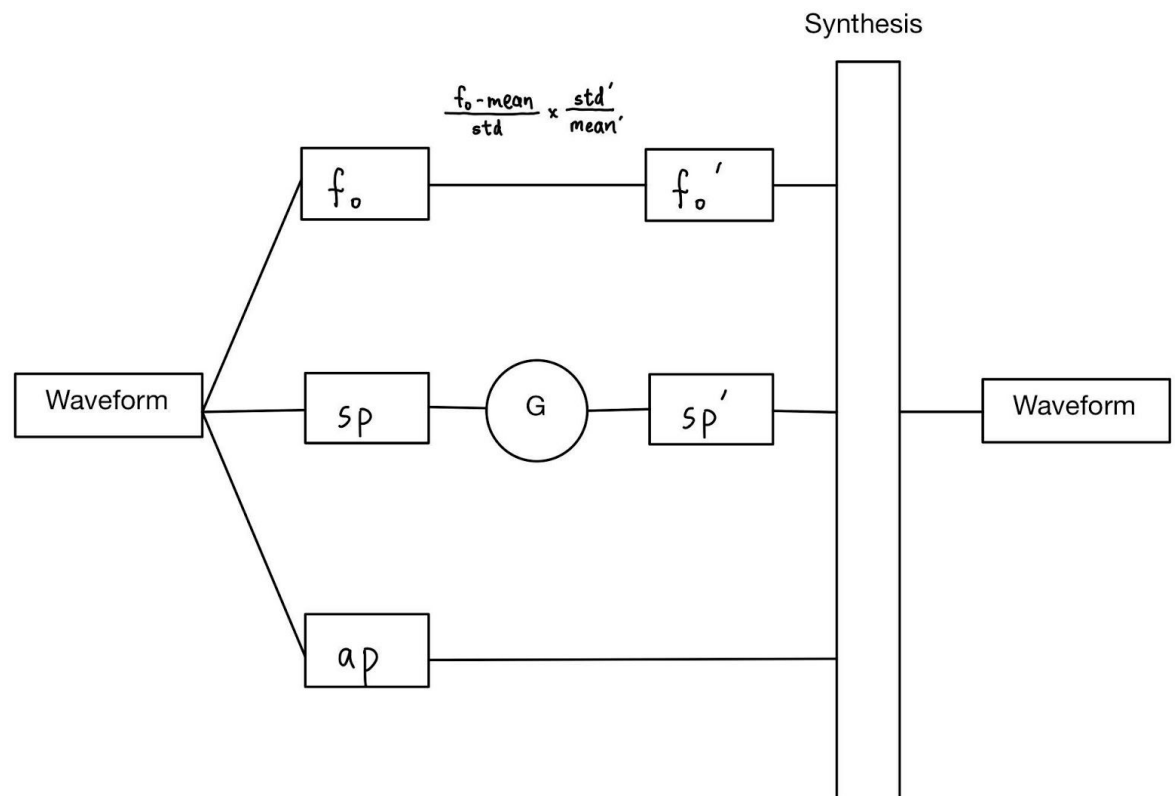
```

103 | self.d1 = Down2d(2, 32, (3,9), (1,1), (1,4))
104 | self.d2 = Down2d(33, 32, (3,8), (1,2), (1,3))
105 | self.d3 = Down2d(33, 32, (3,8), (1,2), (1,3))
106 | self.d4 = Down2d(33, 32, (3,6), (1,2), (1,2))
107 |
108 | self.conv = nn.Conv2d(33, 1, (36,5), (36,1), (0,2))

```

(3) 請問這個程式碼中，input acoustic feature 以及 generator output 分別是什麼呢？(1%) Hint: 請研究一下 preprocess 時做了哪些事情。

input acoustic feature 使用 WORLD 所計算的 mel-cepstral coefficients (fundamental frequency(F0), code spectral envelop(sp), Aperiodic parameter(ap)), generator input 為512個frame的spectral envelop, output亦為如此，最後會將這些sp decode，並將F0 mean和std轉換到所需要target的f0 mean和std，最後在將decoded sp、converted f0、ap 用WORLD做synthesize成最後的輸出的音檔。

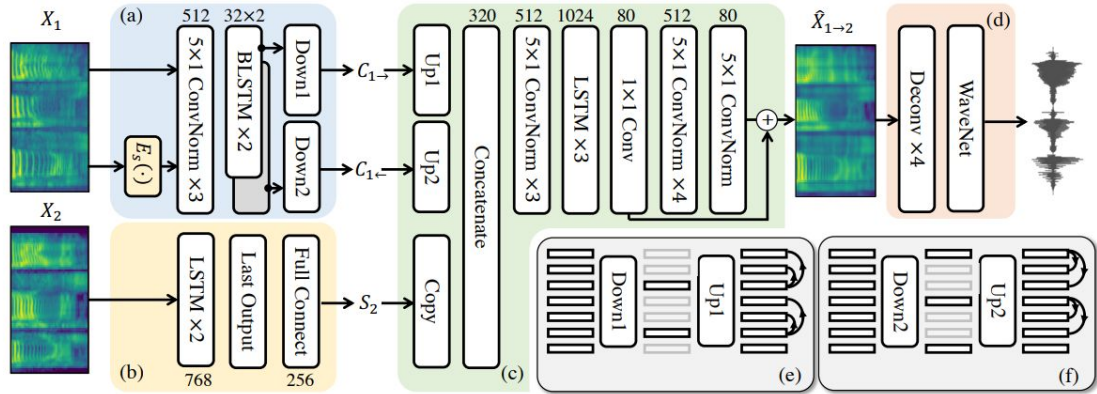


HW2-3 (1) 和 (2) 擇一回答 (4%)

(1) 請自己找一個不是 StarGAN-VC，也不是 HW2-1 的 model，實際 train 看看。請詳細描述 model 得架構，training objective，訓練時是否需要 paired data 等等。(4%) Hint: [useful link](#)

AutoVC(<https://arxiv.org/abs/1905.05879>): 使用 AutoVC 作者提供的 code ,
然後我們有稍微修改, vocoder 是用 universal vocoding
(<https://github.com/bshall/UniversalVocoding>)

模型架構 :



(a) Content Encoder $E_c(\cdot)$, input為80維的mel-spectrogram , 和 speaker embedding 串接 , 經過三個有batch normalize的conv layer , 以及Relu activation , 再用兩個BLSTM產生兩個32維的output(forward和backward) , 將其downsample , forward取time={0, 32, 63...} , backward取{31, 62,...} , 成 $C_{1 \rightarrow}$ 和 $C_{1 \leftarrow}$ 。

(b) Speaker Encoder $E_s(\cdot)$, 使用pre-trained d-vector

(c) Decoder , 將 $C_{1 \rightarrow}$ 和 $C_{1 \leftarrow}$ upsample , 與speaker embedding concat , 經過三層有batch normalize的conv layer , 以及Relu activation , 以及三層LSTM , 再利用1 x 1 conv layer project 到 80 維成

$$\tilde{X}_{1 \rightarrow 2}$$

最後使用post-network , 由五層conv layer組成 , 最後一層降維至80維 , 最後output 為 residual signal , final conversion為

$$\hat{X}_{1 \rightarrow 2} = \tilde{X}_{1 \rightarrow 2} + R_{1 \rightarrow 2}$$

(d) Spectrogram inverter , 使用universal vocoding

training data:

hw2-1 unpaired data (90% train, 10% val) --> 80-dimensional mel spectrogram

training objective:

$$\min_{E_c(\cdot), D(\cdot, \cdot)} L = L_{\text{recon}} + \mu L_{\text{recon0}} + \lambda L_{\text{content}}$$

$$L_{\text{recon0}} = \mathbb{E}[\|\tilde{X}_{1 \rightarrow 1} - X_1\|_2^2]$$

$$L_{\text{recon}} = \mathbb{E}[\|\hat{X}_{1 \rightarrow 1} - X_1\|_2^2]$$

$$L_{\text{content}} = \mathbb{E}[\|E_c(\hat{X}_{1 \rightarrow 1}) - C_1\|_1]$$

訓練50萬epoch，最後 L_{recon} 、 L_{recon0} 、 L_{content} 收斂至0.0004、0.0003、0.0001

成果比較:

因為使用不同的 vocoder 的關係，AutoVC的結果比較少機械音的情況，且聲音比較接近真正對方的聲音，而不像hw2-1的結果只聽出來是男轉女或女轉男<；因為使用pre-trained d-vector，所以可以轉換unseen speaker；但在content的部份，感覺有一些短音被丟掉，造成有些詞有點連起來，猜測可能是中間content encoder output的元素太少，導致表達content的成份不夠所導致，或是因為只使用hw2-1的data，資料量太少導致，但因為時間的緣故，沒有實際更動再train一次，可能之後在分享會時分享是否為這些原因導致。