DLHLP HW2 Voice Conversion Report

組長 github id: wubinary
組員：徐均筑 吳彬睿

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%)**

   encoder 跟 decoder 的架構基本上沒有更動，我們只有把 hidden size 調小，從 512 調到 128，因為實驗發現如果 hidden size 太大，speaker 的 global information會被藏到 latent 裡面，而不是我們想要的 content embedding，因此我們最後選擇了一個適合這個 task 的 hidden size。接著我們將 encoder 跟 decoder 合為一個 auto-encoder 的 module。loss function 的部分，我們使用老師上課講解的 scale invariant signal noise ratio 的去做優化：把 output 投影到原本 input 的 spectrogram上接著再計算 L2 loss。

   我們以 batch size 為 64 去進行 training，大概 3 個 epoch (約 25000 steps) 就收斂了。

```
259 class Autoencoder(nn.Module):
260     def __init__(self):
261         super(Autoencoder, self).__init__()
262         self.encoder = Encoder()
263         self.decoder = Decoder()
264 
265     def forward(self, x, c):
266         latent = self.encoder(x)
267         output = self.decoder(latent, c)
268         return latent, output
269 
270     def interpolate(self, x, c1, c2):
271         latent = self.encoder(x)
272         output = self.decoder.interpolate(latent, c1, c2)
273         return output
```

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%)**

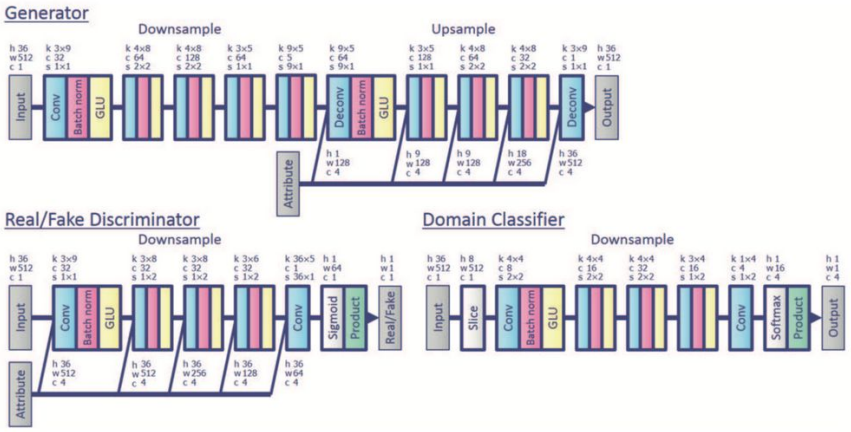
   2_1_338_interpolate: 感覺女性高頻跟男性低頻的聲音還在，反而是比較中間頻率的聲音感覺消失了，所以聽起來有點像是女生開頭男生結尾的發話方式，而語調感覺跟原本的男性還蠻像的。
   1_2_334_interpolate: 感覺原本的 334 音檔的女性聲音保留了很一大部分，雖然是做interpolate的結果，但是感覺不太出來有男性的聲音成分在裡面。
   同時 interpolate 的結果感覺比直接 p1 轉 p2 或是 p2 轉 p1 的結果相較之下雜音好像比較多一點點，而 interpolate 結果的音頻則介於原本p1跟p2的音頻之間，不過還是有稍微偏向女性的聲音的趨勢。

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%)**

   在這個 repo 中，speaker 以 one-hot vector 的形式放入。這個做法的優點在於簡單，但缺點卻是當有新的 speaker 時無法直接以現有 model 套用，另外 one-hot vector 也不含有 speaker 的聲音特性，比如性別、年齡等。除了 one-hot vector 之外，speaker information 也可用 speaker embedding 的方式放入，例如 i-vector, d-vector, x-vector 等，這樣的做法能彌補 one-hot vector 的缺點。

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%)**



   將輸入的 class(attribute) vector 改為 binary，因此 network 中 attribute 的 c 維度改為 1，且計算 loss 時改以 Binary Cross Entropy 計算。

   以下為程式碼更改的部分：

```python
class Generator(nn.Module):
    """docstring for Generator."""
    def __init__(self):
        super(Generator, self).__init__()
        self.downsample = nn.Sequential(
            Down2d(1, 32, (3,9), (1,1), (1,4)),
            Down2d(32, 64, (4,8), (2,2), (1,3)),
            Down2d(64, 128, (4,8), (2,2), (1,3)),
            Down2d(128, 64, (3,5), (1,1), (1,2)),
            Down2d(64, 5, (9,5), (9,1), (1,2))
        )

        ## Modified
        in_channel = 5 + cls_dim
        self.up1 = Up2d(in_channel, 64, (9,5), (9,1), (0,2))
        in_channel = 64 + cls_dim
        self.up2 = Up2d(in_channel, 128, (3,5), (1,1), (1,2))
        in_channel = 128 + cls_dim
        self.up3 = Up2d(in_channel, 64, (4,8), (2,2), (1,3))
        in_channel = 64 + cls_dim
        self.up4 = Up2d(in_channel, 32, (4,8), (2,2), (1,3))

        in_channel = 32 + cls_dim
        self.deconv = nn.ConvTranspose2d(in_channel, 1, (3,9), (1,1), (1,4))
```

```python
class Discriminator(nn.Module):
    """docstring for Discriminator."""
    def __init__(self):
        super(Discriminator, self).__init__()

        ## Modified
        in_channel = 1 + cls_dim
        self.d1 = Down2d(in_channel, 32, (3,9), (1,1), (1,4))
        in_channel = 32 + cls_dim
        self.d2 = Down2d(in_channel, 32, (3,8), (1,2), (1,3))
        self.d3 = Down2d(in_channel, 32, (3,8), (1,2), (1,3))
        self.d4 = Down2d(in_channel, 32, (3,6), (1,2), (1,2))

        self.conv = nn.Conv2d(in_channel, 1, (36,5), (36,1), (0,2))
        self.pool = nn.AvgPool2d((1,64))
```

```python
class DomainClassifier(nn.Module):
    """docstring for DomainClassifier."""
    def __init__(self):
        super(DomainClassifier, self).__init__()
        ## Modified
        self.main = nn.Sequential(
            Down2d(1, 8, (4,4), (2,2), (5,1)),
            Down2d(8, 16, (4,4), (2,2), (1,1)),
            Down2d(16, 32, (4,4), (2,2), (0,1)),
            Down2d(32, 16, (3,4), (1,2), (1,1)),
            nn.Conv2d(16, cls_dim, (1,4), (1,2), (0,1)),
            nn.AvgPool2d((1,16)),
            nn.LogSoftmax()
        )
```

```python
# ================================================================================ #
#                          2. Train the discriminator                              #
# ================================================================================ #
# Compute loss with real audio frame.
## Modified
CELoss = nn.BCELoss()
m = nn.Sigmoid()
cls_real = self.C(x_real).squeeze(1)
cls_loss_real = CELoss(input=m(cls_real), target=speaker_idx_org)
```

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。

Preprocess: convert the wav waveform to mcep feature
在 preprocess 時做了下列的事情：

1) Estimate F0 trajectory
2) Calculate the aperiodicity
3) Calculate the spectrogram
4) Code the spectral envelope given the spectrogram

input acoustic feature 為 (4)，而 generator output 也是。另外 preprocess 時所計算的參數將會用於將 generator output decode 成 waveform。

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 架構， training objective，訓練時是否需要 paired data 等等。(4%)**

   我們使用了 [1] 之中的 variational autoencoding Wasserstein generative adversarial-network (VAW-GAN)，透過 VAE 的 encoder 得到 phonetic content vector (z)，也就是聲音訊號的內容，再將 speaker representation vector (y) 和 z 輸入 decoder(generator) 轉換，利用不同的 y 輸入 generator，就能達到 voice conversion。

   在 training 時，我們利用下圖中的公式作為 objective function，前兩項是用來訓練 VAE，後兩項用於 adversarial learning， 由於後兩項分別為來自 target 的音訊以及合成音訊輸入 discriminator 的期望值，因此不需要 alignment，使用 unparallel data 即可。

$$
\begin{aligned}
J_{vawgan} = & -\mathcal{D}_{\mathrm{KL}}\left(q_\phi(\boldsymbol{z}_n|\boldsymbol{x}_n)\|p(\boldsymbol{z}_n)\right) \\
& + \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_\theta(\boldsymbol{x}|\boldsymbol{z},\boldsymbol{y})\right] \\
& + \alpha\,\mathbb{E}_{\boldsymbol{x}\sim p_t^*}\left[\mathcal{D}_\psi(\boldsymbol{x})\right] \\
& - \alpha\,\mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\mathcal{D}_\psi(\mathcal{G}_\theta(\boldsymbol{z},\boldsymbol{y}_t))\right]
\end{aligned}
\tag{13}
$$

where α is a coefficient which emphasizes the W-GAN loss. This objective is shared across all three components: the encoder, the synthesizer, and the discriminator. The synthesizer minimizes this loss whereas the discriminator maximizes it;

   我們利用 HW2-1中的data進行訓練，得到的結果相較於單純的 autoencoder 有明顯的進步，轉換後的音訊較為自然。和 autoencoder 相比，VAW-GAN 利用了 VAE 以及 adverserial learning，另外相較於 VAE-GAN，VAW-GAN 使用 Wasserstein objective，discriminator 的目的不是用於分辨 true/fake。

Reference

[1] C.-C. Hsu, H.-T. Hwang, Y.-C. Wu, Y. Tsao, and H.-M.Wang, “Voice conversion from unaligned corpora using variational autoencoding Wasserstein generative adversarial networks,” inProc. INTERSPEECH, 2017, pp. 3164–3168.