DLHLP HW2 Voice Conversion Report

組長 github id: shuoenchang
學號：R08922A02 系級： AI所碩一 姓名：張碩恩
學號：R08922130 系級： 資工所碩一 姓名：丁杰

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

(1) 請以 Auto-Encoder 之方法實做 Voice conversion。jo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，果餓未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%)

A: 我使用的模型是助教提供的程式碼，因為作業只需要用到AE的部分，所以在訓練的時候，我將原本程式碼除了訓練AE以外的code都註解掉，在原本的程式碼中是沒有儲存AE訓練好的模型，所以另外加了存模型的程式碼，每五萬步存一次模型。

```
if args.train:
    solver.train(args.output_model_path, args.flag, mode='pretrain_G')
    #solver.train(args.output_model_path, args.flag, mode='pretrain_D')
    #solver.train(args.output_model_path, args.flag, mode='train')
    #solver.train(args.output_model_path, args.flag, mode='patchGAN')
```

dingjie

```
if iteration == 1 :
    self.save_model(model_path, iteration)
if iteration % 50000 == 0 :
    self.save_model(model_path, iteration)
```

在convert階段的時候，因為原本程式碼會加上generator所產生的東西，在這部分因為我們沒有訓練generator所以需要將它去掉。

```
for utt_id in f_h5[f'{dset}/{src_speaker}']:
    sp = f_h5[f'{dset}/{src_speaker}/{utt_id}/lin'][()]
    converted_sp = convert_sp(sp, speaker2id[tar_speaker], solver, gen=False)
    wav_data = sp2wav(converted_sp)
    wav_path = os.path.join(dir_path, f'{src_speaker}_{tar_speaker}_{utt_id}.wav')
    wavfile.write(wav_path, 16000, wav_data)
    c += 1
    if c >= max_n:
        break
```

(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%)

A: 我interpolation的方法是將原本的音檔和轉換後的音檔做一個weighted的相加。

在做interpolation之後，會有一個混合音的感覺。
音高的部分在原檔和轉換成異性聲音後的檔案之間，因為男女生的聲音本來就一低一高，所以很正常男女混合後的音高在中間的位置。
口音部分應該就是為什麼聽起來有混合音的原因，因為一個是原本性別的口音，另一個則是轉換後的異性口音，當兩者混在一起會有一點像兩者一起講話又像是同一個人講話的樣子。
語調的部分，因為轉換只需要轉換成異性，原本的語調會保留下來，所以轉換前跟轉換後的語調會差不多，所以interpolation的結果也會是差不多的語調。

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%)

A: 他是以新增 N 個 channel，其中屬於該語者的 channel 全部放 1，其餘為 0 的方式將語者資訊放入模型中。換句話說，也就是將語者的 one-hot encoding 沿著 channel 的方向 concat 到 feature map 中。
這樣做的好處是很直覺，如果想要解釋模型時，也可以很明確知道不同的 channel 中的 parameter 各自的作用。然而，一改變 attributes 數量，整個模型就必須要做更動，更不用說是重新訓練了。
其他的方法舉兩個不同的例子。

1. 另外訓練一個語者資訊的 embedding，並利用這個 embedding 告知到模型語者資訊。
2. 使用 conditional instance normalization[1]，在 CV 領域的 style transfer 有使用過這種方法，在 2-3 的時候我也會嘗試用這種方法做看看。

(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%)

A: 在 dataloader 中，用原先的程式跑的話，label 會被轉換成 binary 的形式，必須手動將其轉為 one-hot encoding 的形式。

```
27  27          filename = os.path.basename(p)
28  28          speaker = filename.split(sep='_', maxsplit=1)[0]
29  29          label = self.encoder.transform([speaker])[0]
    30  +       if   label==[0]:  label=[1, 0]
    31  +       elif label==[1]:  label=[0, 1]
30  32          mcep = np.load(p)
31  33          mcep = torch.FloatTensor(mcep)
32  34          mcep = torch.unsqueeze(mcep, 0)
```

在 model 的部分也必須做更動，原始的模型有 4 個 attributes，但現在剩下 2 個，所以模型中牽扯到 attributes channel 的部分都需要更動。

[1] V. Dumoulin, J. Shlens, and M. Kudlur. A learned representation for artistic style. International Conference of Learned Representations (ICLR), 2016.

```
65        -         self.up1 = Up2d(9, 64, (9,5), (9,1), (0,2))
66        -         self.up2 = Up2d(68, 128, (3,5), (1,1), (1,2))
67        -         self.up3 = Up2d(132, 64, (4,8), (2,2), (1,3))
68        -         self.up4 = Up2d(68, 32, (4,8), (2,2), (1,3))
      65  +         self.up1 = Up2d(7, 64, (9,5), (9,1), (0,2))
      66  +         self.up2 = Up2d(66, 128, (3,5), (1,1), (1,2))
      67  +         self.up3 = Up2d(130, 64, (4,8), (2,2), (1,3))
      68  +         self.up4 = Up2d(66, 34, (4,8), (2,2), (1,3))
69    69
70    70            self.deconv = nn.ConvTranspose2d(36, 1, (3,9), (1,1), (1,4))
71    71
      @@ -100,16 +100,15 @@ class Discriminator(nn.Module):
100   100       def __init__(self):
101   101           super(Discriminator, self).__init__()
102   102
103       -         self.d1 = Down2d(5, 32, (3,9), (1,1), (1,4))
104       -         self.d2 = Down2d(36, 32, (3,8), (1,2), (1,3))
105       -         self.d3 = Down2d(36, 32, (3,8), (1,2), (1,3))
106       -         self.d4 = Down2d(36, 32, (3,6), (1,2), (1,2))
      103 +         self.d1 = Down2d(3, 32, (3,9), (1,1), (1,4))
      104 +         self.d2 = Down2d(34, 32, (3,8), (1,2), (1,3))
      105 +         self.d3 = Down2d(34, 32, (3,8), (1,2), (1,3))
      106 +         self.d4 = Down2d(34, 32, (3,6), (1,2), (1,2))
107   107
108       -         self.conv = nn.Conv2d(36, 1, (36,5), (36,1), (0,2))
      108 +         self.conv = nn.Conv2d(34, 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。

A: acoustic feature 是 mcep feature，語音的資料經過轉換到 mcep feature 後，會以圖片的形式保存，如此就能將做為 starGAN 的 input。
而在 GAN 中 generator 其實就是要產生跟 input 類似的檔案，所以產生的結果也會是圖片形式的 mcep feature，得到這些 feature 後再靠語音合成的技術就可以還原出聲音資訊。

HW2-3 (1) 和 (2) 擇一回答 (4%)

Q: 想辦法 improve HW2-1或是 HW2-2 的 model (或是改一些有趣的東西)。Hint: 各位可以想想看 speaker embedding 有沒有什麼其他方式？如果今天我在 testing 的時候想要讓他有 unseen speaker 也可以成功轉過去的話，用什麼 embedding會比較好？(hint: d-vector, i-vector) 又或者要怎麼把這個 speaker embedding 餵進 model 裡面呢？有什麼不同的方法？

A: 這部分主要參考 StarGAN-VC2[2] 中的分析，來改善 2-2 在 loss function 上的問題。除了 loss 部份外，我還改變了語者嵌入模型的方式。在實作的細節，我還有把 2-2 實作時使用到的 GAN 改成 WGAN，並參考 CycleGAN-VC2[3]，在 generator 的 downsample 和 upsample 之間加入 1D residual block。綜合以上

[2] Kaneko, T., Kameoka, H., Tanaka, K., & Hojo, N. StarGAN-VC2: Rethinking Conditional Methods for StarGAN-Based Voice Conversion. *ArXiv, abs/1907.12279*, 2019

[3] Kaneko, T., Kameoka, H., Tanaka, K., & Hojo, N. Cyclegan-VC2: Improved Cyclegan-based Non-parallel Voice Conversion. *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 6820-6824, 2019

4 項改善得到了更好的結果。以下稍微講述 StarGAN-VC2 中提到的 loss 問題是什麼。

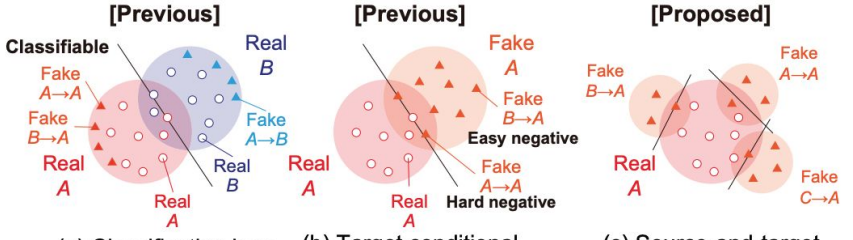
在 StarGAN 中，主要使用 classification loss 和 target conditional adversarial loss。

1. figure a：在 classification loss 中，黑色的 boundary 線是在 real-data domains 學出來的，而 Generator 要做的是產生出可以被分類到其他類的資料，如果以資料分佈來看，他會偏向產生遠離 boundary 線的資料 (easily “classifiable” data)，如此以來將會導致 Generator 無法 cover all real data distribution。
2. figure b：在 target conditional adversarial loss 中，我們希望找到 real data 和 fake data 間的 boundary。我們可以避免 generator 只產生出了 classifiable data，但卻與 real data 有差距很大的結果。然而這個 loss 在處理不同的狀況時會有不太平衡的問題，如要判別 hard negative (conversion between the same speaker ex,A → A) 和判別 easy negative (conversion between different speakers ex,B → A) 的難度不一樣。
3. figure c：基於以上的理由，這篇論文提出了 source-and-target conditional adversarial loss。

$$\mathcal{L}_{t\text{-}adv} = \mathbb{E}_{(\boldsymbol{x},c)\sim P(\boldsymbol{x},c)}[\log D(\boldsymbol{x}, c)] + \mathbb{E}_{\boldsymbol{x}\sim P(\boldsymbol{x}),c'\sim P(c')}[\log(1 - D(G(\boldsymbol{x}, c'), c'))] \quad \text{(starGAN-VC)}$$

$$\mathcal{L}_{st\text{-}adv} = \mathbb{E}_{(\boldsymbol{x},c)\sim P(\boldsymbol{x},c),c'\sim P(c')}[\log D(\boldsymbol{x}, c', c)] + \mathbb{E}_{(\boldsymbol{x},c)\sim P(\boldsymbol{x},c),c'\sim P(c')}[\log D(G(\boldsymbol{x}, c, c'), c, c')] \quad \text{(starGAN-VC2)}$$

可以看出跟原本 loss 的差別在於在 generator 和 discriminator 上皆使用了 source c 和 target c’，如此相較原先的 loss 多了更多限制，彼此產生互相制衡的作用，改善了前面提到的 loss 在不同狀況下不平衡的問題，如圖描述就比較不會有 easy negative 或 hard negative 的問題。另一方面來說，discriminator 可以同時判別 target class，也就不需要 classification loss 了。對 generate 出來的結果，也可以讓其更靠近 real data 的狀況。



(a) Classification loss (b) Target conditional adversarial loss (c) Source-and-target conditional adversarial loss

---

除了以上參考 starGAN VC loss 外的另一個改變，是改善語者嵌入模型的方式，我試著把模型從 channel-wise 的方式改變成 conditional instance normalization 的方式，作法的表示式是

$$\mathrm{CIN}(x; s) = \gamma^s \left(\frac{x - \mu(x)}{\sigma(x)} \right) + \beta^s$$

其中s代表不同語者， γ 和 β 是根據不同語者學出來的。訓練出來的 γ 和 β 相較 channel-wise 的方式，如果要應用在其他需要 embedding 的地方，也比較不受限制，而且在訓練時也比較有效率。