DLHLP HW2 Voice Conversion Report

組長 github id: ChuanYouLin
組員：
學號：r08944008 系級：網媒所碩一 姓名：簡　義
學號：r07922104 系級：資工所碩二 姓名：林傳祐
學號：r08944024 系級：網媒所碩一 姓名：陳品媛
學號：r06922089 系級：資工所碩二 姓名：邱淳浩

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%)

ANS:

模型參數的部份

- Voice-Conversion/hps/vctk.json
  - `"enc_pretrain_iters": 100000`
- Voice-Conversion/hps/en_speaker_used.txt
  - 改為僅使用speaker 1, 2

程式更改的部份

- Voice-Conversion/main.py
  - 只訓練pretrain_G，註解掉其他的mode

```
solver = Solver(hps_tuple, data_loader)
if args.load_model:
    solver.load_model(args.load_model_path)
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')
```

- Voice-Conversion/convert.py
  - 測試speaker改為1, 2
  - convert_all_sp的gen參數設為False，關掉微調用的generater

```python
if __name__ == '__main__':
    h5_path = './vctk.h5'
    root_dir = './results'
    model_path = './pkl/model.pkl'
    hps_path = './hps/vctk.json'
    solver = get_model(hps_path = hps_path, model_path = model_path)
    speakers = ['1', '2']
    max_n = 5
    if len(sys.argv) == 3:
        speakers = speakers[:min(5, int(sys.argv[1]))]
        max_n = min(5, int(sys.argv[2]))
    for speaker_A in speakers:
        for speaker_B in speakers:
            if speaker_A != speaker_B:
                dir_path = os.path.join(root_dir, f'p{speaker_A}_p{speaker_B}')
                if not os.path.exists(dir_path):
                    os.makedirs(dir_path)
                convert_all_sp(h5_path, speaker_A, speaker_B,
                               solver, dir_path, gen = False,
                               max_n = max_n)
```

(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%)

ANS:

將兩個speaker的embedding做內插，interpolate ratio設定為0.5，即0.5*p1+0.5*p2。觀察音檔334與338後，發現兩份都會有interpolated的聲音與原先的source speaker比較像的現象，也就是說如果p1轉p1與p2的內插，聽起來會像p1的聲音。

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%)

ANS:

語者資訊是直接和content資訊做concat，concat方法為將語者資訊的維度作為channel和content做concat。

e.g. (spk_dim, h, w) 和 (channel, h, w) concat，結果為 (spk_dim + channel, h, w)。

這樣嵌入語者資訊的方法，當需要增加speaker時，會造成model需要的參數變多。

若使用ivector或dvector等可以表達語者的feature，可以處理one-hot encoding語者數量不同會造成feature vector大小會不同的問題。語者資訊加入generator

也可以採用直接和content feature做運算，而不要使用concat的方式，可以減少model參數量的增長。

(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%)

ANS:

由於speaker information的embedding維度指有1 (兩個speaker，[0] or [1])，所以Generator和Discriminator network中的channel size要做調整：

```
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))
        )


        # self.up1 = Up2d(9, 64, (9,5), (9,1), (0,2))
        self.up1 = Up2d(6, 64, (9,5), (9,1), (0,2))
        # self.up2 = Up2d(68, 128, (3,5), (1,1), (1,2))
        self.up2 = Up2d(65, 128, (3,5), (1,1), (1,2))
        # self.up3 = Up2d(132, 64, (4,8), (2,2), (1,3))
        self.up3 = Up2d(129, 64, (4,8), (2,2), (1,3))
        # self.up4 = Up2d(68, 32, (4,8), (2,2), (1,3))
        self.up4 = Up2d(65, 32, (4,8), (2,2), (1,3))

        # self.deconv = nn.ConvTranspose2d(36, 1, (3,9), (1,1), (1,4))
        self.deconv = nn.ConvTranspose2d(33, 1, (3,9), (1,1), (1,4))
```

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

        # self.d1 = Down2d(5, 32, (3,9), (1,1), (1,4))
        self.d1 = Down2d(2, 32, (3,9), (1,1), (1,4))
        # self.d2 = Down2d(36, 32, (3,8), (1,2), (1,3))
        self.d2 = Down2d(33, 32, (3,8), (1,2), (1,3))
        # self.d3 = Down2d(36, 32, (3,8), (1,2), (1,3))
        self.d3 = Down2d(33, 32, (3,8), (1,2), (1,3))
        # self.d4 = Down2d(36, 32, (3,6), (1,2), (1,2))
        self.d4 = Down2d(33, 32, (3,6), (1,2), (1,2))

        # self.conv = nn.Conv2d(36, 1, (36,5), (36,1), (0,2))
        self.conv = nn.Conv2d(33, 1, (36,5), (36,1), (0,2))
        self.pool = nn.AvgPool2d((1,64))
```

由於Speaker只有兩個，因此DomainClassifier的最後convolution的輸出channel改成2，並且由於CrossEntropyLoss會做LogSoftmax，因此拿掉最後的LogSoftmax。

```
class DomainClassifier(nn.Module):
    """docstring for DomainClassifier."""
    def __init__(self):
        super(DomainClassifier, self).__init__()
        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, 4, (1,4), (1,2), (0,1)),
            nn.Conv2d(16, 2, (1,4), (1,2), (0,1)),
            nn.AvgPool2d((1,16)),
            # nn.LogSoftmax()
        )
```

(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。

ANS:

這個work使用的vocoder是World Vocoder(https://github.com/JeremyCCHsu/Python-Wrapper-for-World-Vocoder)，該vocoder共使用了三種feature - spectral envelope, logarithmic fundamental frequency, aperiodicities，model的input feature是其中的spectral envelope再經過處理得到的mcep，而generator的output是mcep，test時的mecp有做normalized。

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

(2) 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？

ANS:

修改 HW2-1的auto-encoder，分別使用了pre-trained的 d-vector, x-vector取代原本的embedding。透過pre-trained model取到的256維d-vector經由Linear層轉至512維，使其可以與原先的feature相加與進行相關的運算。而x-vector的部份，由於本身的維度已是512維，所以直接送進model沒有多做處理。上述兩個

vector是NN train語者辨識時的bottleneck feature，因此已經包含了語者資訊，理論上在test時，unseen speaker的資訊也可以透過d-vector或x-vector獲得，在轉unseen speaker上應該也有一定的效果。測試時使用google小姐作為target，但由於現在training只使用了2個speaker，資料量不足以讓model對unseen speaker有足夠的generalization，因此針對unseen speaker的效果不太好。

reference:

d-vector: https://github.com/resemble-ai/Resemblyzer/tree/master

x-vector: https://github.com/zeroQiaoba/ivector-xvector