

DLHLP HW2 Voice Conversion Report

組長 github id: PengWenChen

組員：趙達軒、許博閔、陳玠玟、陳芃玟

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，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%)

由於題目其中一個要求為 combine two speaker 的聲音，因此我們僅修改了 decoder 的部份，將 decoder 原本的 speaker embedding 改成相加除以二。

```
406 def forward2(self, x, c1, c2):
407     # conv layer
408     out = self.conv_block(x, [self.conv1, self.conv2], self.ins_norm1, (self.emb1(c1)+self.emb1(c2))/2, res=True)
409     out = self.conv_block(out, [self.conv3, self.conv4], self.ins_norm2, (self.emb2(c1)+self.emb2(c2))/2, res=True)
410     out = self.conv_block(out, [self.conv5, self.conv6], self.ins_norm3, (self.emb3(c1)+self.emb3(c2))/2, res=True)
411     # dense layer
412     out = self.dense_block(out, (self.emb4(c1)+self.emb4(c2))/2, [self.dense1, self.dense2], self.ins_norm4, res=True)
413     out = self.dense_block(out, (self.emb4(c1)+self.emb4(c2))/2, [self.dense3, self.dense4], self.ins_norm5, res=True)
414     emb = (self.emb5(c1)+self.emb5(c2))/2
415     out_add = out + emb.view(emb.size(0), emb.size(1), 1)
416     # rnn layer
417     out_rnn = RNN(out_add, self.RNN)
418     out = torch.cat([out, out_rnn], dim=1)
419     out = append_emb((self.emb5(c1)+self.emb5(c2))/2, out.size(2), out)
420     out = linear(out, self.dense5)
421     out = F.leaky_relu(out, negative_slope=self.ns)
422     out = linear(out, self.linear)
423     #out = torch.tanh(out)
424     return out
```

- (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%)

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%)

連結中的方法將語者資訊轉成 one-hot vector，以 github 題供之範例為例：4 個 speaker 的資訊會以 1*4 的 one-hot vector 表示（若 target 為第 3 個語者，vector 中第3個值為1 其餘為 0）。one-hot vector 會有無法任意增減語者的問

題，若想從 4 個與者轉換改成 2 個語者轉換模型就要重新訓練。speaker embedding 還有其他方式，例如 i-vector, d-vector, x-vector... 等等。

- (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%)

由於 speaker 從 4 個變成 2 個，所以我們將 domain classifier 的 output 從維度為 4 的 vector 修改成維度 1。然後將 domain classifier 的 loss 改成 BCEWithLogitsLoss。

- (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。

這個程式碼的 input acoustic feature 為 Mel Cepstral Coefficients (MCEP)，相較於 MFCC，MCEP 可以保留比較多的語音資訊，因此做語音合成時時常使用 MCEP 作為 acoustic feature。Generator 的 output 也是 acoustic feature，比較不一樣的是 generator 在 decode 時 concat target label。

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](#)

在這題中，我們參考了助教給的link，也就是TTS without T。垃圾

cVAE:

我們直接使用了 2-1 Encoder 和 Decoder 架構，並修改 Encoder 的部份。

```

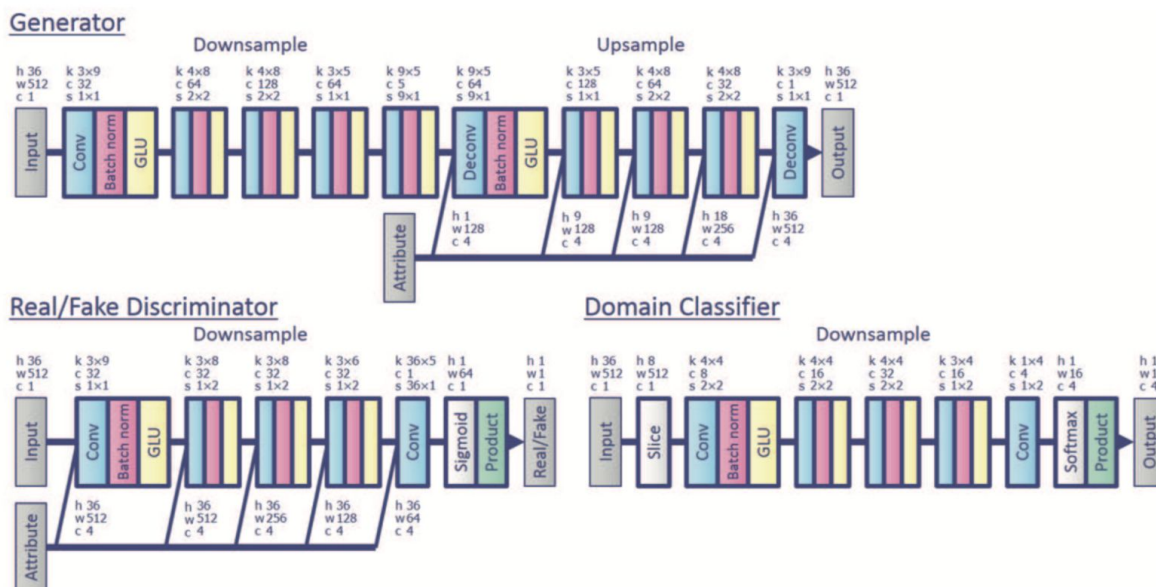
590 .....def reparametrize(self, mu, logvar, train=True):
591 .....    if train:
592 .....        std = torch.exp(0.5*logvar)
593 .....        eps = torch.randn_like(std)
594 .....        return eps*std + mu
595 .....    else:
596 .....        return mu
597
598 .....def forward(self, x):
599 .....    aa=c.view(c.size(0), 1, 1)
600 .....    bb=aa.repeat(1, 1, x.size(2))
601 .....    x=torch.cat([x, bb.float()], 1)
602 .....    outs=[]
603
604 .....    for l in self.convs:
605 .....        out=pad_layer(x, l)
606 .....        outs.append(out)
607
608 .....    out=torch.cat(outs+[x], dim=1)
609 .....    out=torch.cat([out, bb], dim=1)
610 .....    out=F.leaky_relu(out, negative_slope=self.ns)
611 .....    out=self.conv_block(out, [self.conv2], [self.ins_norm1, self.drop1], res=False)
612 .....    out=self.conv_block(out, [self.conv3, self.conv4], [self.ins_norm2, self.drop2])
613 .....    out=self.conv_block(out, [self.conv5, self.conv6], [self.ins_norm3, self.drop3])
614 .....    out=self.conv_block(out, [self.conv7, self.conv8], [self.ins_norm4, self.drop4])
615 .....    #dense layer
616 .....    out=self.dense_block(out, [self.dense1, self.dense2], [self.ins_norm5, self.drop5], res=True)
617 .....    out=self.dense_block(out, [self.dense3, self.dense4], [self.ins_norm6, self.drop6], res=True)
618 .....    out_rnn=RNN(out, self.RNN)
619 .....    out=torch.cat([out, out_rnn], dim=1)
620 .....    mu=linear(out, self.linear1)
621 .....    var=linear(out, self.linear2)
622 .....
623 .....    mu=F.leaky_relu(mu, negative_slope=self.ns)
624 .....    var=F.leaky_relu(var, negative_slope=self.ns)
625 .....    mu=out[:, 0:512, :]
626 .....    var=out[:, 512:, :]
627 .....    output=self.reparametrize(mu, var)

```

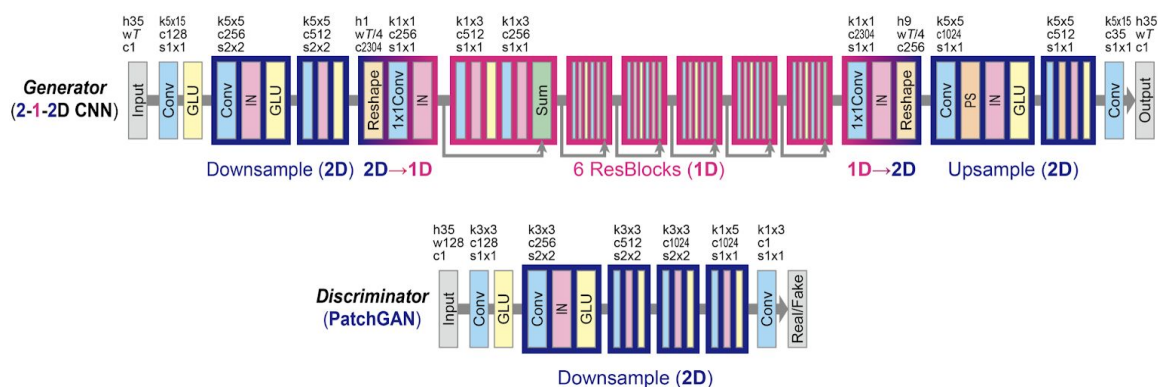
我們在 Encoder 的地方也加入了語者資訊，並最後改為輸出 mean 跟 std。

- (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？

CycleGAN-VC2:



圖（一）：hw2-2的架構

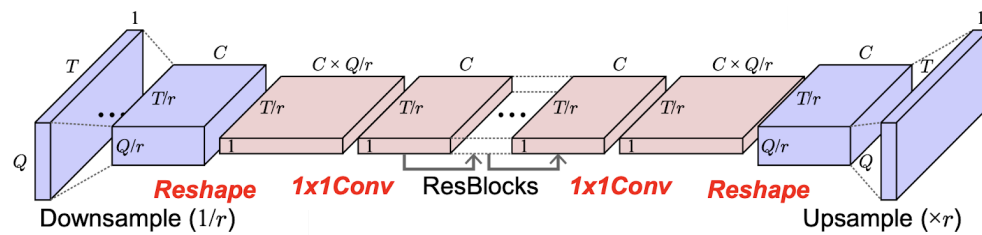


圖（二）：hw2-3的架構

這個方法是根據hw2-2的model做修改，因為hw2-2只用了兩個語者做訓練，基本上可以看作是一個CycleGAN。

這個方法跟hw2-2的StarGAN（見圖一）差別在於Generator具有2D-1D-2D CNN結構，而不是基於2D CNN的結構。

在這個網絡（見圖二）中，將2D卷積用於下採樣和上採樣，並將1D卷積用於主轉換過程（即residual blocks），並用1x1 convolution調整channel數目（見圖三）。



圖（三）：2D-1D-2D CNN結構

最後，將domain classifier拿掉。

```

156 class Generator(nn.Module):
157     def __init__(self):
158         super(Generator, self).__init__()
159
160         self.conv1 = nn.Conv2d(in_channels=1,
161                                out_channels=128,
162                                kernel_size=[5,15],
163                                stride=1,
164                                padding=[2,7])
165
166         self.conv1_gates = nn.Conv2d(in_channels=1,
167                                       out_channels=128,
168                                       kernel_size=[5,15],
169                                       stride=1,
170                                       padding=[2,7])
171
172         # Downsample Layer
173         self.downSample1 = downSample_Generator(in_channels=128,
174                                                  out_channels=256,
175                                                  kernel_size=5,
176                                                  stride=2,
177                                                  padding=2)
178
179         self.downSample2 = downSample_Generator(in_channels=256,
180                                                  out_channels=512,
181                                                  kernel_size=5,
182                                                  stride=2,
183                                                  padding=2)
184
185         #reshape
186         self.conv2 = nn.Conv1d(in_channels=3072,
187                                out_channels=512,
188                                kernel_size=1,
189                                stride=1)

```