语音转换 VC 小结

黄圣杰

2020 年 3 月 6 日

目录

1 语音转换续集

1.1 语音信号处理总体架构

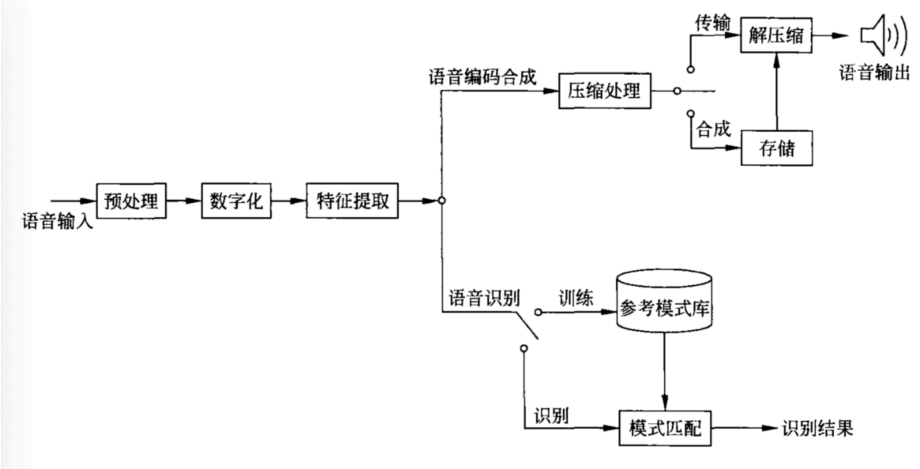

图 1: 语音信号处理的总体结构框图

从这个总体结构可以看出：无论是语音识别，还是语音编码与合成，输入的语音信号首先要进行预处理。

预处理 ：对信号进行适当放大和增益控制，并进行反混叠滤波来消除工频信号的干扰。

数字化 ：将模拟信号转化为数字信号便于用计算机来处理。

提取特征：用反映语音信号特点的若干参数来代表语音。(常见的比如：F0，MCC，mecp ……)

语音转换：多篇论文说到，可以将 VC 任务看做是“音色转换任务”的分支

1.2 语音转换导论

这里可以查看之前的一份知乎上王赟 (yun) 博士的一篇综述 知乎 ，文章从语音转换课题从最早期的发展和演变都阐述得较为接地气，这里先不做赘述。

总的上来说，语音转换领域在钻研的问题主要有这么几类：

1. **语料不平行问题**；(暂时不做具体过细解释，需要开源时再补充细节)

有两种类型的语音转换数据。这是并行数据（源 speaker 和目标 speaker 说话内容一致，paired data）和非并行数据（源 speaker 和目标 speaker 的说话内容不一致，unpaired data）。

• 使用并行数据的语音转换使用：

发出相同语音内容的两种类型的语音作为训练数据。因为话语的内容是相同的，以可以转换声学特征而不用担心语言特征。但是，有必要使用动态时间扭曲 DTW）等进行对齐，因为说话单词的时间由于说话速度的不同而发生偏移。在这种情况下，语音转换在某些情况下可能并不容易，因为在考虑数据集的准备时必须使用相同的单词。

• 使用非并行数据进行语音转换：

一般是指，采集到的数据中，源说话人和目标说话人的说话内容不一致，更接近普适的情景，并在此条件下，希望生成一段语音，其内容由原说话人决定，而声学特征（语气、断句、韵律等）由目标说话人决定。在这种情况下，不需要对齐，并且比并行数据更容易收集数据集。

2. **能否实现多对多的问题**；（many-to-many）

3. **声音“塌缩”问题**；（collapse，即体现在声音听起来沉闷，甚至不像源 & 目标）

4. **少数据集的问题**；（这一点通常是以上几个问题都会伴随的难点问题）

上面几点，在近几年的前沿文章角度来看，平行语料的转换基本没有问题，甚至在少数据集的条件下，也能相对完美地转换。（这点也在旭东学长与陌陌 HR 交流中，也得到了印证）

在接下来，我将各类文章按照技术路线/模型结构角度来进行做一个小结。有总结不到位、不清楚的地方还请读者多多斧正！

2 几种技术路线

2.1 Cycle-GAN

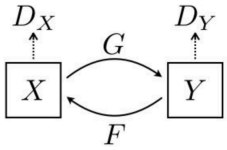

图 2: CycleGAN 示意图

最早的一篇 CycleGAN-VC 论文[2]，是在 2017 年发表的，这篇也是王赟博士在综述的 4.1 中“生成对抗式网络”小节里面的最近一篇。但我们在上一段时间针对 VC 领域论文的全方位阅读中发现，近两年在 CycleGAN 这条路线上，已经有了更长足的进步和发展演变。

CycleGAN-VC 使用 mel cepstrums（MCEP）来表示来自各种可能的语音特征的人类语音颜色。论文使用 WORLD 来提取 24 维 MCEP 并将其用作网络的输入。

最后，损失函数类似于 CycleGAN。它使用对抗性损失，Cycle 一致性损失，身份映射损失。在论文中，声称最后的身份映射损失保留了语言结构。

以下先简单从 CycleGAN-VC 的三个损失函数角度来阐述一下的运行原理:

1. **对抗损失**

为了使转换后的特征与目标 y 难以区分，使用对抗性损失:

$$\begin{aligned}\mathcal{L}_{adv}\left(G_{X\to Y}, D_Y\right) = & E_{y\sim P_Y(y)}\left[\log D_Y(y)\right] \\ & + E_{x\sim P_X(x)}\left[\log\left(1 - D_Y\left(G_{X\to Y}(x)\right)\right)\right]\end{aligned} \tag{1}$$

CycleGAN 中除了两个生成器以外，还有两个判别器，用于分辨两个生成器的输出与两个说话人的真实语音。注意，这里不需要生成器的输出与真实语音对齐，连平行数据都不需要！

判别器 D 试图通过最大化这种损失来寻找实数和转换特征之间的最佳决策边界，而生成器 G 试图通过最小化这种损失来生成可以欺骗的特征。

这和正常的 GAN 思路一致，并以此过程中，相互轮流冻结 & 训练参数，最终实现 G & D 两者的共同进步。G 能生成更逼真的特征，D 能有更强的分辨真假特征的能力。

2. **循环一致性损失**

对抗性损失只限制 $G_{X\to Y}(x)$ 服从目标分布，不保证输入输出特征的语言一致性。为了进一步规范映射，使用循环一致性损失:

$$\begin{aligned}&\mathcal{L}_{cyc}\left(G_{X\to Y}, G_{Y\to X}\right) \\ =&E_{x\sim P_X(x)}\left[\left\|G_{Y\to X}\left(G_{X\to Y}(x)\right) - x\right\|_1\right] \\ +&E_{y\sim P_Y(y)}\left[\left\|G_{X\to Y}\left(G_{Y\to X}(y)\right) - y\right\|_1\right]\end{aligned} \tag{2}$$

同时学习正反向映射以稳定训练。这种损失促使 $G_{X\to Y}, G_{Y\to X}$ 通过循环变换找到 (X, Y) 的最优伪对。如图 3 a 所示:

GAN 有一个臭名昭著的缺陷，就是在训练过程中容易发生 mode collapsing，通俗点儿说就是不管输入是什么，都给出一样的输出。循环一致性损失的设置，就是为了防止 mode collapsing，让两个生成器能够保持语音的内容。其具体做法是，要求语音在 CycleGAN 中转一圈后，与初始时基本一致，即 $F(G(x))$ 尽可能接近 x ，$G(F(y))$ 尽可能接近 y 。如果发生了 mode collapsing，在一个方向上损失了的信息是不可能在另一个方向找回来

的。

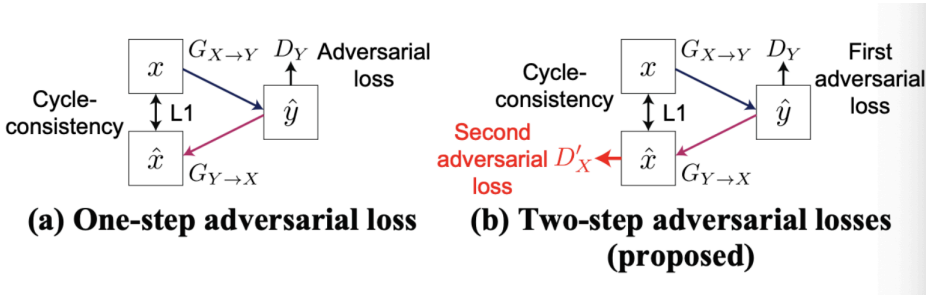

图 3: CycleGAN 循环一致性损失

3. **全等映射损失**

「全等映射损失」又称作「身份映射损失」，用来进一步鼓励语言信息的保存：

$$\begin{aligned}\mathcal{L}_{id}\left(G_{X\to Y}, G_{Y\to X}\right) = E_{y\sim P_Y(y)}\left[\|G_{X\to Y}(y) - y\|_1\right] \\ + E_{x\sim P_X(x)}\left[\|G_{Y\to X}(x) - x\|_1\right]\end{aligned} \tag{3}$$

即使使用了循环一致性损失，CycleGAN 里的生成器还是有可能对语音的内容做手脚。打个极端的比方，两个生成器除了转换说话人的身份之外，还把语音的内容变成了原来的反义句，这样的 CycleGAN 同样可以让循环一致性损失很小，因为语音内容取了两次反之后又变回了原样。全等映射损失，就是让 $G(y)$ 尽可能接近 y 、$F(x)$ 尽可能接近 x ，即当 G 的输入已经是目标说话人的语音、F 的输入已经是源说话人的语音时，两个生成器应当充当一个全等映射，不要对语音的内容做任何的改动。

综上，整体 CycleGAN 的 Loss 为：

$$\begin{aligned}\mathcal{L}_{\text{full}} = \mathcal{L}_{adv}\left(G_{X\to Y}, D_Y\right) + \mathcal{L}_{adv}\left(G_{Y\to X}, D_X\right) \\ + \lambda_{cyc}\mathcal{L}_{cyc}\left(G_{X\to Y}, G_{Y\to X}\right) + \lambda_{id}\mathcal{L}_{id}\left(G_{X\to Y}, G_{Y\to X}\right)\end{aligned} \tag{4}$$

其中 λ_{cyc} 和 λ_{id} 为权衡参数。在这个公式中，每个周期使用一次对抗性损失，如图 3 a 所示。因此，我们称其为一步对抗性损失。

接下来再复习一下 CycleGAN 的网络结构：

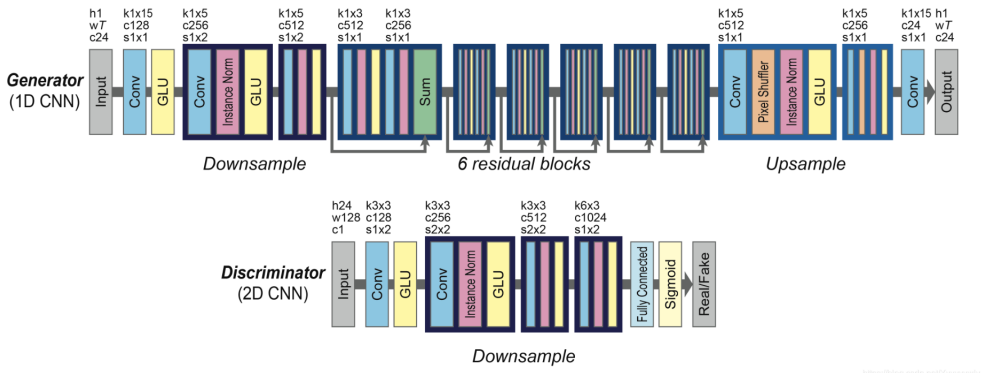

图 4: CycleGAN 网络结构

• 观察生成器和鉴别器的网络结构。在输入或输出层中，h，w 和 c 分别表示通道的高度，宽度和数量。在每个卷积层中，k，c 和 s 分别表示内核大小通道数和步幅大小。由于发生器是完全卷积的，它可以输入任意长度 T。

1 • Instance Norm 层：可以理解为对数据做一个归一化的操作。

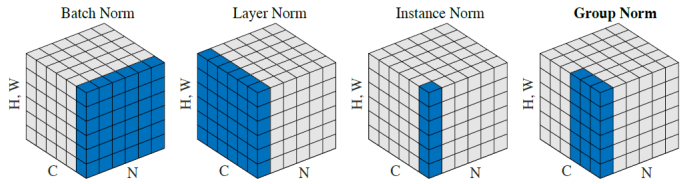

Figure 2. **Normalization methods**. Each subplot shows a feature map tensor, with N as the batch axis, C as the channel axis, and (H, W) as the spatial axes. The pixels in blue are normalized by the same mean and variance, computed by aggregating the values of these pixels.

图 5: IN 示意图

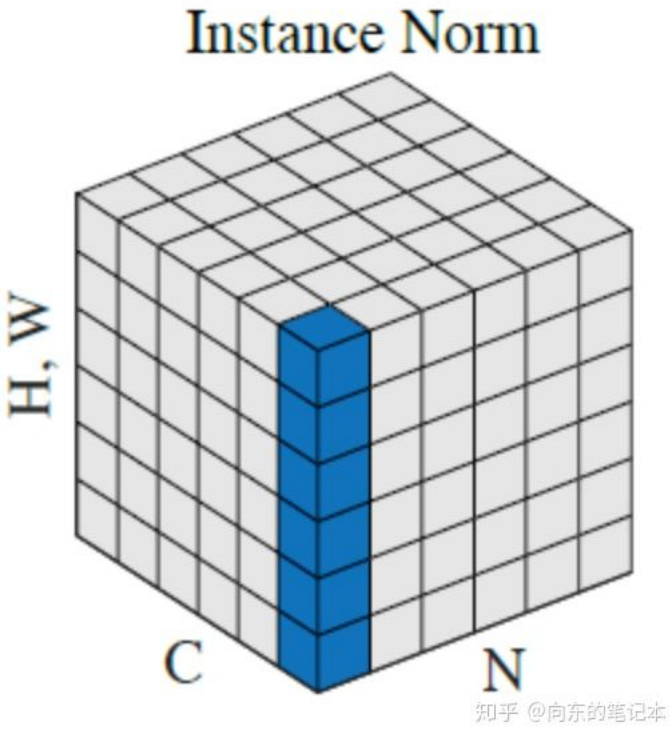

图 6: IN 小示意图

主要的几种归一化方法如图5所示，咱们重点看一下 Batch & IN 的区别：这里先要做一个假设，读者都和我一样复习过了 CNN 中的操作，即卷积的流程和模型数字代表的意思。看6，假如现有 6 张图片，每张图片在 CNN 的某一卷积层有 6 个通道，也就是 6 个 feature map。从 C 角度看过去，是一个个 channel 通道，从 N 方向看过去，是一张张图片，每 6 个竖着排列的小正方体组成的长方体代表一张图片的一个 feature map。蓝色的方块是一起进行 Normalization 的部分。

由此就可以很清楚的看出，Batch Normalization 是指 6 张图片中的每一张图片的同一个通道一起进行 Normalization 操作。而 Instance Normalization 指单张图片的单个通道单独进行 Noramlization 操作。

IN 适用于生成模型中，比如图片风格迁移。因为图片生成的结果主要依赖于某个图像实例，所以对整个 batch 归一化不适合图像风格化中，在风格迁移中使用 Instance Normalization 不仅可以加速模型收敛，并且可以保持每个图像实例之间的独立。类比到语音转换中，就可以理解为，在语音风格迁移工作时，加速音频的模型收敛并且保持每个音频实例之间的独立。（其实有点不太懂这个独立在风格转换中有什么作用）

$$y_{tijk} = \frac{x_{tijk} - \mu_{ti}}{\sqrt{\sigma_{ti}^2 + \epsilon}}, \quad \mu_{ti} = \frac{1}{HW}\sum_{l=1}^{W}\sum_{m=1}^{H} x_{tilm}, \quad \sigma_{ti}^2 = \frac{1}{HW}\sum_{l=1}^{W}\sum_{m=1}^{H}(x_{tilm} - mu_{ti})^2.$$

图 7: IN 公式

IN 公式中，t 代表图片的 Index，即 N 方向看过去，i 代表的是 feature map 的 index，即六个蓝色政法提组成的整体。

IN 算法过程：

- 沿着通道计算每张图的均值 u
- 沿着通道计算每张图的方差 σ^2
- 对 x 做归一化，$x' = (x - u)/\sqrt{(\sigma^2 + \varepsilon)}$
- 加入缩放和平移变量 γ 和 β，归一化后的值，$y = \gamma x' + \beta$

注释：以上内容参考学习了 知乎内容 讲的非常好！

---

2 • GLU 是一个数据驱动的激活函数，并且门控机制允许根据先前的层状态选择性地传播信息。（为什么要用激活函数？如果不用激励函数，每一层输出都是上层输入的线性函数，无论神经网络有多少层，输出都是输入的线性组合。如果使用的话，激活函数给神经元引入了非线性因素，使得神经网络可以任意逼近任何非线性函数，这样神经网络就可以应用到众多的非线性模型中。）

若要问激活函数是什么，那就理解成在神经网络中，将上一层的输出

& 下一层的输入，之间的数据，做一个非线性的转换。如果没有非线性的激励，那么中间的隐藏层就失去意义，这种情况就成了最原始版本的“感知机”(Perceptron)；那么网络的逼近能力就有限。这才又了非线性的各种激励函数，理论上可以逼近任意函数。(即神经网络能学习任何可参数化的东西)

若要说起 GLU，可以顺带提一下激活函数家族史：深度学习神经网络中最先被广泛使用的激活函数是 Sigmoid 函数和双曲正切函数 tanh，都是非线性的激活函数。

- sigmoid 函数：$f(x) = 1/(1 + exp(-x))$
- 双曲正切函数：$f(x) = tanh(x)$

他们有两个明显缺点：

1. 存在饱和死区，所以反向传播时会导致梯度消失 & 梯度爆炸；
2. 复杂神经网络关于参数 W 的梯度计算过于复杂；

之后引入了 Relu(Rectified linear unit) 及系列变体来解决上面提到的两个问题：

Relu 激活函数的表达式为：

$$f(x) = max(0, x)$$

再之后，借鉴了 LSTM 的 Gate Mechanism 思想，基于 Relu 和 tanh 激活函数，结合 gate unit ，产生出来了 GTU units & GLU units 等激活单元。

- GTU（Gated Tanh Unit）的表达式为：

$$f(X) = tanh(X * W + b) * O(X * V + c)$$

组成结构：Tanh 激活单元：$tanh(X * W + b)$，加上一个 Sigmoid 激活单元：$O(X * V + c)$，构成的 gate unit，就构成了 GTU 单元。

• GLU（Gated Liner Unit）表达式为：

$$f(X) = (X * W + b) * O(X * V + c)$$

$$h_l(\mathbf{X}) = (\mathbf{X} * \mathbf{W} + \mathbf{b}) \otimes \sigma(\mathbf{X} * \mathbf{V} + \mathbf{c}) \quad (1)$$

where $\mathbf{X} \in \mathbb{R}^{N \times m}$ is the input of layer h_l, that is either word embeddings or the outputs of previous layers, $\mathbf{W} \in \mathbb{R}^{k \times m \times n}$, $\mathbf{b} \in \mathbb{R}^n$, $\mathbf{V} \in \mathbb{R}^{k \times m \times n}$, $\mathbf{c} \in \mathbb{R}^n$ are learned parameters, σ is the sigmoid function and $\otimes$ is the element-wise product between matrices.

图 8: GLU

组成结构：Relu 激活单元：$(X * W + b)$，加上一个 Sigmoid 激活单元：$O(X * V + c)$，构成的 gate unit，就构成了 GLU 单元。

其中其中 W 和 V 是不同的卷积核，卷积核宽度为 k，输出通道数为 n，b 和 c 是偏置参数。

上面公式中的后半部分，即有 sigmoid 激活函数的卷积就是所谓的门控机制，其控制了 $X * W + b$ 中哪些信息可以传入下一层。这里将其定义为 Gated Linear Units (GLU). 然后就可以将该模型进行堆叠，以捕获 Long-Term memory。

我们知道，sigmoid 函数输出区间为 [0,1], 所以人们通常用信息流经过 sigmoid 之后，得到的数值来生动地描述“门”这个概念。0 就表示“关闭”，1 就表示“开启”，在 [0,1] 之间，就表示部分信息可传递下去，另一部分就无法通过。理解了就行，我这里见的资料显示，一般都是采用 sigmod 模式来模拟“门控”，详细的在下面“门控 CNN ”里面讲述。

---

3 • 门控 CNN 思想：模拟 lstm 的是否遗忘门，或者说判断权重的思想。再做一个和 CNN 卷积一样参数的 filter，取值 0-1，判断这个序列的特征哪些应该被关注，哪些应该被忽略。

想理解好“门控”的概念，需要简单理解一下 LSTM 中引入的经典门控，参考这篇 博客，大意是说，lstm 引入了三个门：输入门、输出门、遗忘门。结合上 LSTM 具有汲取过往信息的能力，“门”这个概念就更像是一个水龙头，其中的参数决定了每一个环节，输出 & 接受输入的程度有多少，并以这些“门”中的参数来进一步增强整个网络的精度，弥补之前的激活函数的不足，或者说增强之前的激活函数。注：LSTM 门控是结合 sigmoid 来实现，跟上一小节说的内容呼应。

更具体地来说，在 Gate CNN（门控 CNN ）中，与传统 CNN 的区别只在于卷积层，将原本没有经过非线性输出的部分再乘上经过 sigmoid 非线性变换的部分，以此得到一个输出值，传递给下一层。参看图8; 所谓的门限卷积，其核心在于为卷积的激活值添加一个门限开关，来决定其有多大的概率传到下一层去。

与 RNN 逐个处理输入序列不同，CNN 可以实现并行计算，大大加快训练速度。并且分层结构也简化了学习，与 RNN 的链结构相比，非线性计算数量减少，从而减轻了消失梯度问题。值得一提的就是卷积中不需要和 LSTM 那样使用“遗忘门”，更具体的请参看 [1]. 以上内容参考了一些博客

以上内容有些啰嗦，但是也是很基础很重要的入门知识。

再来了解一下 CNN 中处理的是什么? 根据论文 [3] 所述，CNN 处理的是光谱纹理(spectral texture) -> Mel-cepstral 系数(MCEP)(梅尔倒谱系数)

• CycleGAN-VC 使用一维 (1D) CNN 作为生成器，在保留时间结构的同时，捕捉整个关系和特征方向。这可以看作是逐帧模型的直接时间扩展，该模型只捕获每帧的这些特性的关系。为了在保持输入结构的同时有效地捕获大范围的时间结构，该生成器由下采样层、残差层和上采样层组成，图 4a 所示。另一个值得注意的地方是 CycleGAN-VC 使用一个门控 CNN 来捕捉声学特征的顺序和层次结构。

• CycleGAN-VC 使用 2D CNN 作为鉴别器来聚焦于 2D 结构 (即 2D 光谱纹理)。更具体的，如图 4b 所示，考虑到输入的整体结构，在最后一

层使用全连接层「Full Connected」来确定真实感。这样的模型称为 FullGAN。

综上所述，用 CycleGAN 进行语音转换，打破了「需要平行训练数据」的限制，但源和目标说话人的身份仍然需要事先指定。还不能做到 many-to-many，但在同时代的作品中，算是跨出了平行数据的限制，并且取得了 baseline 级别的效果。

2.1.1 CycleGan & CycleGAN 2

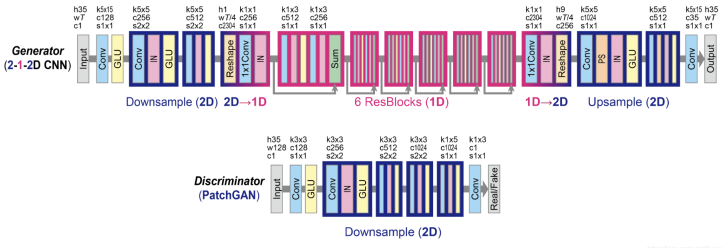

图 9: CycleGAN-VC2

CycleGAN-VC2，它是 CycleGAN-VC 的改进版本，包含三种新技术：改进的目标（两步对抗性损失），改进的发生器（2-1-2D CNN））和改进的判别器（Patch GAN）。

$$\begin{aligned}\mathcal{L}_{adv2}\left(G_{X\to Y}, G_{Y\to X}, D'_X\right) = E_{x\sim P_X(x)}\left[\log D'_X(x)\right] \\ +E_{x\sim P_X(x)}\left[\log\left(1 - D'_X\left(G_{Y\to X}\left(G_{X\to Y}(x)\right)\right)\right)\right]\end{aligned} \tag{5}$$

2.1.2 CycleGan & StarGAN

2.1.3 StarGAN & StarGAN 2

2.2 Auto-Encoder 框架

2.2.1 VAE-VC

2.2.2 Cycle-VAE

2.2.3 VAE-GAN

2.2.4 AutoVC-ZeroShot

2.3 VAE 变种

2.3.1 CDVAE

2.3.2 CDVAE+CLS+GAN

VAE——>CDVAE——>CDVAE+CLS （CLS 带 gan 训练）+GAN

2.4 VCC2018 效果最好框架

2.4.1 关于 N10 作品

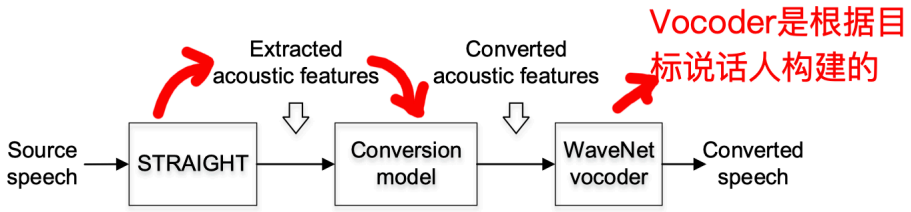

Figure 1: *The process of voice conversion using a WaveNet vocoder.*

图 10: N10 流程图

说实话，这么多论文看过来，N10[4] 虽然效果最好，但是无奈我理解能力有限，而且文章没有像平常文章那样说的很具体详细，所以理解起来

有不少困难。这里我查了一些博客资料，引用别人的一段话来总结这份文章：

首先用带有对齐转录的数百个小时的外部语音数据训练一个提取器；然后从 source 中提取 conten posterior feature，将这个特征送入说话者依赖的 LSTM-RNN 网络中预测 F0 和 STRAIGHT 谱特征，最后用 wavenet speaker-dependent 输出重建语音波形。训练阶段有一些人工的检查和数据的标注，包括 F0 提取误差的校正、移除一些不规则发声段。

可以补充的一小点就是，这份方案，采取了如10所示的流程，他在特征处理环节（前端）采用了 STRAIGHT 这种传统的语音编码器，作用上就和后面 N17 内容可以看到的 WORLD 一样，用来提取各种语音特征。

2.4.2 关于 N17 作品——平行实验

怕遗忘了最重要的 N17&N10 作品模型，先写下理解的内容：

N17 平行组的论文 [5] 所采用的方法和 N10 有相似之处，最明显的就是采用 wavenet 结构，

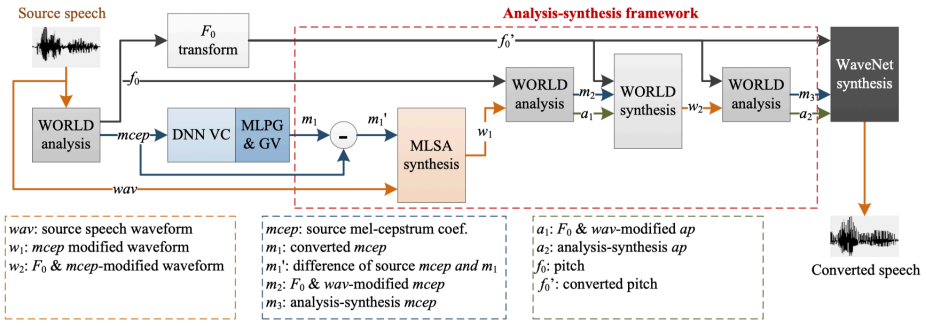

Figure 1. *Basic NU parallel VC system based on DNN and WaveNet vocoder*

图 11: N17 平行整体网络

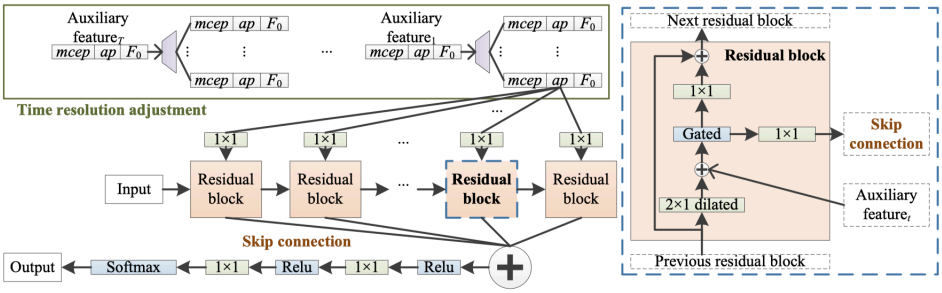

Figure 2. *Conditional WaveNet vocoder architecture*

图 12: N17 平行 wavenet

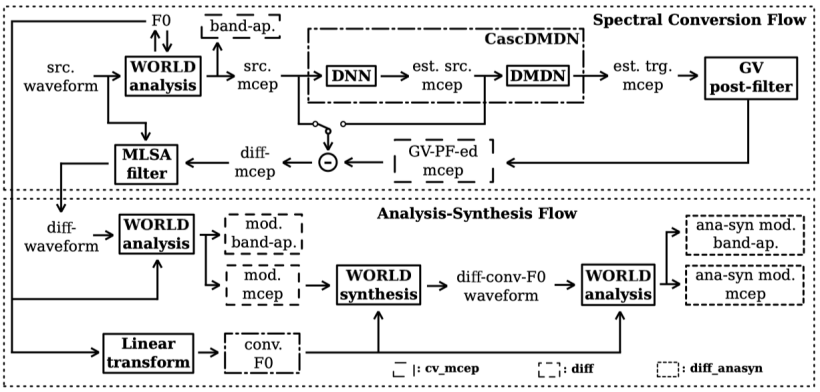

Figure 2: Diagram of the spectral conversion flow (top) and the analysis-synthesis flow (bottom) to generate three different speech parameters, i.e., "cv_mcep", "diff", and "diff_anasyn", to be fed into the waveform-processing module.

图 13: N17 平行具体

关于平行实验的处理流程，在先看完非平行部分的内容之后，可能更好理解一点。先说明一下整体上的处理流程，然后再做进一步思考。正如图11所示，先是原始的语音输入，用 WORLD 分析器，提取出 mcep 和 f_0 特征。一条线上，f_0 进行线性转换处理之后，在之后的每个合成步骤都有用上。第二部分，是将 mecp 特征做 DNN 转换，并再经过 MLPG[7]（maximum likelihood parameter generation）（论文中关于 MLPG 的说明也仅仅只有一个引用文献，但是拿来看，也搞不明白怎么处理），以及 GV[6]（global variance）转换处理之后，与原始语音的 mcep 特征进行一个"做差"的方法，得到 m_1'，之后用 m_1' 和原始语音波形 wav 进行第一次合成，MLSA[8] 合

成：mel log spectrum approximation (MLSA) filter, 得到 w_1 波形，这部分波形是仅仅修改过 mecp 特征的波形。

接着将 w_1 和 f_0 一起再送到下一个 WORLD 分析器，提取出新的 mcep 特征，以及一个新的 ap（aperiodicity features）特征，这两个元素，和之前经过线性转换的 f_0‘，一起送到 WORLD 合成器中去合成语音。注意这里已经用上了第二种合成器。

然后，得到 w_2 波形，和处理 w_1 步骤类似，结合上经过线性转换的 f_0'，还是送进 WORLD 分析器，得出 m_3 这个 mcep 特征和 a_2 这个 ap 特征，最后与 f_0' 一起送到 wavenet 合成器中，合成出最终的声音。

以上是整个流程，12谈到了 wavenet 合成器中的具体执行步骤。在读论文过程中，老师和我们一起在想说，为什么要做这么多个步骤的特征提取以及语音合成，按理说，三个合成步骤，都能产生转换后的波形。确实原本没想清楚，但是现在看过 N17 非平行的实验之后，有了一个简单的猜想，可能这里也是采用了和非平行实验最后一步类似的“挑选”缓解：三个合成器，合成出来的频谱特征差距都不大，但是我们实验指导 wavenet 的听觉效果更自然，所以这里等于将原本设想的并行的三种合成，转化成了线性运行的合成（我在非平行中的猜想是又两个以上的并行合成，并在最后进行 δp 和 δL 的评测，最后根据这两个差值决定采用 wavenet 效果还是传统合成器效果），至于为什么可以直接在一条路线上运行三种合成器，而却不影响太多效果，我猜测，一来是因为我们用了平行的数据，没有了非平行的顾虑。二来，我猜测是因为文中说到传统的语音转换，在波形和频谱上没有过多的改变，主要的变动都是由于 wavenet 自己产生的，所以，这里就用上了三种合成器，并最终用 wavenet 收尾！

2.4.3 关于 N17 作品——非平行实验

论文 [9] 讲述了这么一个流程：为了解决非平行的语音转换问题，尝试在原本非平行的 Source 和 Target 之间，再创造一个 Reference speaker，而不是直接进行 S 和 T 的转换；在训练步骤，将语音看做由两部分组成，一是语言信息，二是说话人信息。如果理解成两个人的声音传递，则先让 S 说

一句话，然后请 R 学着说一遍这句话，R 说话内容和 S 一样，这部分在训练 Source speaker encoder；

之后由 R 和 T 进行对接转换，让 T 说一段和 R 一样的内容，这部分在训练 Target speaker decoder。

在转换步骤，原始语音输入，经过 WORLD 提取出 mecp 特征和 f0 特征，将 mcep 送入前面训练好的转换模型中，以此来得到转换后的 mecp。这里有一个细节，就是文中说到 GV 后置过滤器（查了文献，不知道这个是什么），可以提升人感知的音质，但也会增加误差，所以在转换模型中，只在最后出口使用它。另一条路线，将 f0 做一个线性变换成为 f0'，最后将转换后的 mecp、f0'、语音波形 wav 一起送进分析合成框架 & wavenet 合成器中，得到转换后的语音。

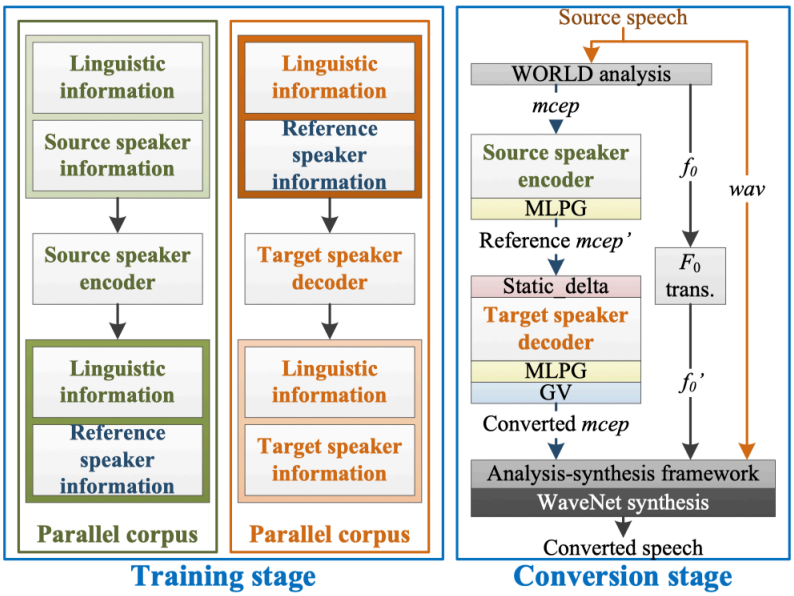

Figure 3. *NU non-parallel VC system*

图 14: N17 非平行

文中又有谈到，虽然 wavenet 的语音合成质量大部分都比传统合成办法更高，但是有时候还是会因为某些特征在训练过程中没匹配好，导致了声

音“塌缩”，听起来转换的不好。

另一方面，通过实验知道，虽然传统的转换方法出来效果不够好，但是两种方法转换的语音，在正确转换的情况下，他们的波形和频谱都相差无几。所以本文想出了一种检测办法，在一条路线上用 wavenet，另一条路线上用传统的转换方法（文章没提具体用什么方法），两者并行开始。然后又设置了两个检测指标，δP 和 δL，分别表示两者的功率谱之差和奈奎斯特频率之差。只要在转换时，这两者的数据，没有超过预先设置的数值，就采用 wavenet 的转换效果。不然就采用传统的转换方法。以此，来解决 wavenet 偶然出错的情况。

3 关于 VCC2016

网上看到一篇质量非常高的关于 VCC2016 的总结，还没来得及细看，先记录一下。

voice conversion 任务主要由两个步骤构成，特征提取与特征参数转换，对于这两个步骤，都有相应的常用的技术，这两个步骤中常用的技术各种排列组合，就产生了众多VC系统，以下做小汇总。

STEP1：Feature extraction			STEP2：Feature conversion
Feature	Extraction toolkits	description	
LSF（Line Spectral Frequency）	STRAIGHT	STRAIGHT is an MATLAB Lib design for VC.	GMM/JDGMM
MGC（mel-generalized coeffi-cient ）	STRAIGHT		DNN
LF0(log f0)	AHOCODER	Ahocoder parameterizes speech waveforms into three different streams	RNN(BLSTM)
MCP(mel-spectgram)	AHOCODER		seq2seq/(with Attention)
MVF（maximum voiced frequency）	AHOCODER		GRU
	DBN		VQ
	Mixture of Factor Analyzer		
parameter generation algorithm with global variance	HTS	HMM-based Speech Synthesis System	

其中,LF0为语音基频log变换，为主要的语音转换参数，是表征不同人，不同性别的最重要参数之一。
其他参数为语音的高阶分量，控制合成语音的细节。

图 15: VCC2016

4 结论

所以综合以上几种作品，咱们已经知道，在语音风格转换中，分为两个大的步骤，一是进行特征转换，以此来体现“内容不变，但是说话人信息改变成目标说话人”。二是，采用 VCC2018 中效果最好的 wavenet 办法，进行声音合成。

对于第一点，做一些可行方案的提出：虽然不是非常明白 N17 的非平行实验中为什么要做那么多次的、多种合成器的语音合成步骤，但是既然可行，更重要的是，这份作品把流程讲得十分清楚！所以不妨先试试能否复现一下 N17 的作品（平行部分）；至于非平行部分，虽然咱们指导文章在做什么，但是相对于平行实验讲得那么具体，非平行部分并没有太详细的操作步骤描述。所以我还是不太懂怎么操作，才能实现论文里说的，将 S 先转换为

中间说话人 R ，然后再转换为目标说话人 T 。

后面的合成器，采用 wavenet，如果让我想策略，我还是会先试着复现 N17 当中说的办法。具体要怎么实现以及根据我们的需求来改进，得集思广益了，目前水平有限，说的不合常理容易闹笑话 hhh。

LaTeX, 我看行!

References

[1] Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks, 2016.

[2] Takuhiro Kaneko and Hirokazu Kameoka. Parallel-data-free voice conversion using cycle-consistent adversarial networks, 2017.

[3] Takuhiro Kaneko, Hirokazu Kameoka, Nobukatsu Hojo, Yusuke Ijima, Kaoru Hiramatsu, and Kunio Kashino. Generative adversarial network-based postfilter for statistical parametric speech synthesis. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4910–4914. IEEE, 2017.

[4] Li-Juan Liu, Zhen-Hua Ling, Yuan Jiang, Ming Zhou, and Li-Rong Dai. Wavenet vocoder with limited training data for voice conversion. In *Interspeech*, pages 1983–1987, 2018.

[5] Patrick Lumban Tobing, Yi-Chiao Wu, Tomoki Hayashi, Kazuhiro Kobayashi, and Tomoki Toda. Nu voice conversion system for the voice conversion challenge 2018. In *Odyssey*, pages 219–226, 2018.

[6] Tomoki Toda, Alan W Black, and Keiichi Tokuda. Voice conversion based on maximum-likelihood estimation of spectral parameter trajectory. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(8):2222–2235, 2007.

[7] Keiichi Tokuda, Takao Kobayashi, and Satoshi Imai. Speech parameter generation from hmm using dynamic features. In *1995 International*

Conference on Acoustics, Speech, and Signal Processing, volume 1, pages 660–663. IEEE, 1995.

[8] Keiichi Tokuda, Takao Kobayashi, Takashi Masuko, and Satoshi Imai. Mel-generalized cepstral analysis-a unified approach to speech spectral estimation. In *Third International Conference on Spoken Language Processing*, 1994.

[9] Yi-Chiao Wu, Patrick Lumban Tobing, Tomoki Hayashi, Kazuhiro Kobayashi, and Tomoki Toda. The nu non-parallel voice conversion system for the voice conversion challenge 2018. In *Odyssey*, pages 211–218, 2018.