语音识别与合成 talkingface-toolkit 小组大作业

1 模型项目基本概述

语音转换 VC 研究集中在保留节奏、语调以及语言内容。为了从源头上保留这些特性，原项目将当前的非并行 VC 系统分解为两个编码器和一个解码器。通过多次实验分析每个模块，并重新组装最佳组件，提出了 Assem-VC，这是一种全新的最先进的任意多对多非并行 VC 系统。此外，PPG 和 Cotatron 特性依赖于说话者，因此该项目尝试通过对抗性训练去除说话者身份。

具体地，代码和音频样本可在 https://github.com/mindslab-ai/assem-vc 获得。本次作业旨在将原始项目的代码重构到 talkingface-toolkit 框架并能够正常完成训练和推理等功能。

2 环境配置与依赖库

本项目作业通过 Conda 实现 Python 虚拟环境的构建，版本为 3.6.8。对于依赖库，可根据 requirements.txt 文件进行安装，具体路径见 ./requirements.txt，其中指定了各个库的名称和对应版本，支持在 Terminal 进入环境下使用 pip install 命令安装。

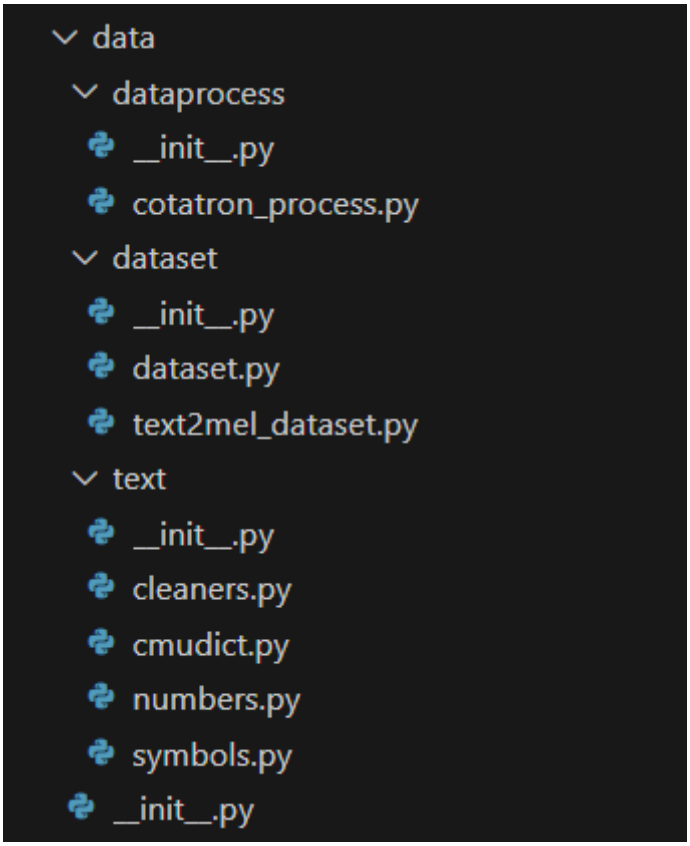
3 重构功能与实现

根据原始项目以及新的 talkingface 框架架构可见，项目具体地分为数据、模型与训练两个部分便于说明。

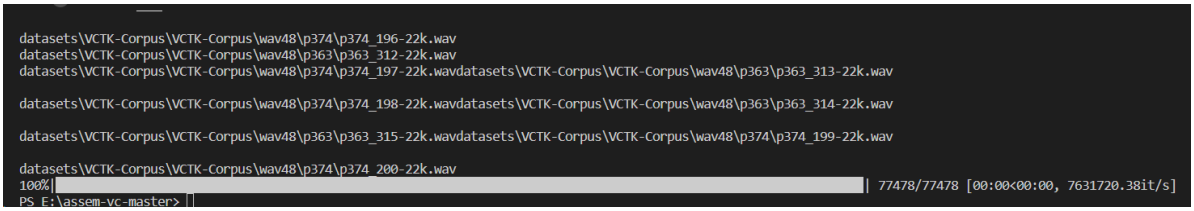
3.1 数据

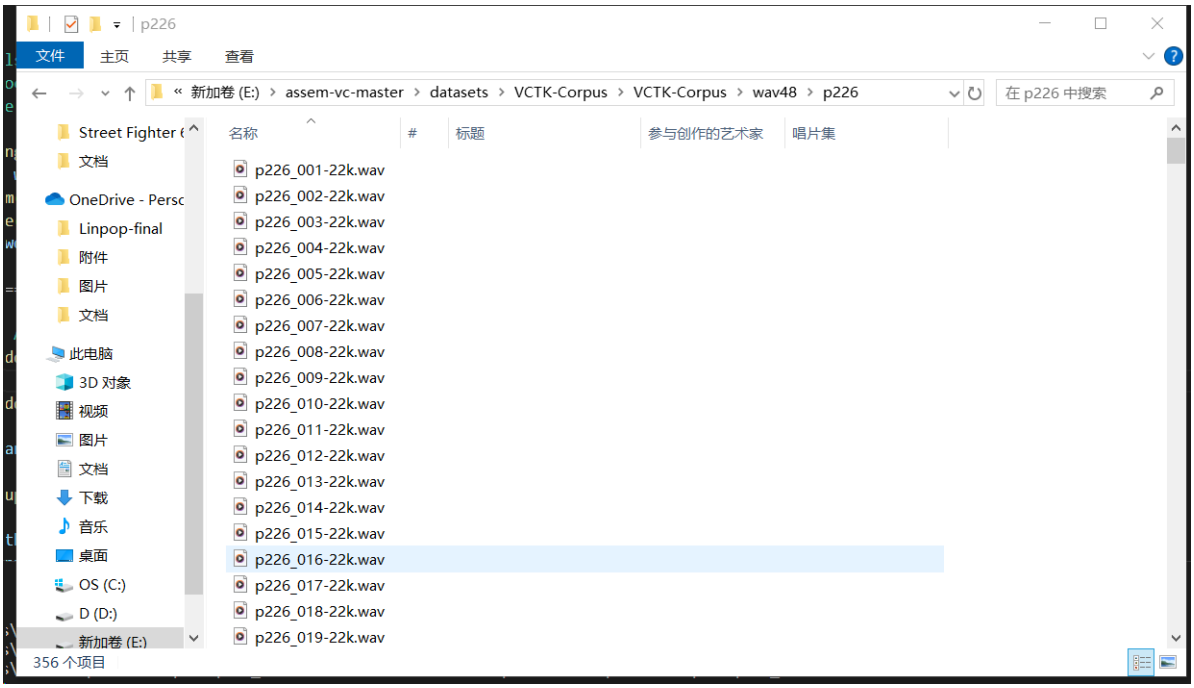
项目使用的数据集为 LibriTTS train-clean-100 和 VCTK dataset 0.8.0，它们所对应的获取网络地址是 http://www.openslr.org/resources/60/train-clean-100.tar.gz 和 https://datashare.ed.ac.uk/handle/10283/2651。数据集的主要数据格式为 .wav 文件，存放在 dataset 文件夹下，其原始内容需要经过特定的预处理步骤，为人声音频的描述，具体实验中划分为 train、test、val 三个部分以完成基本的训练需求。

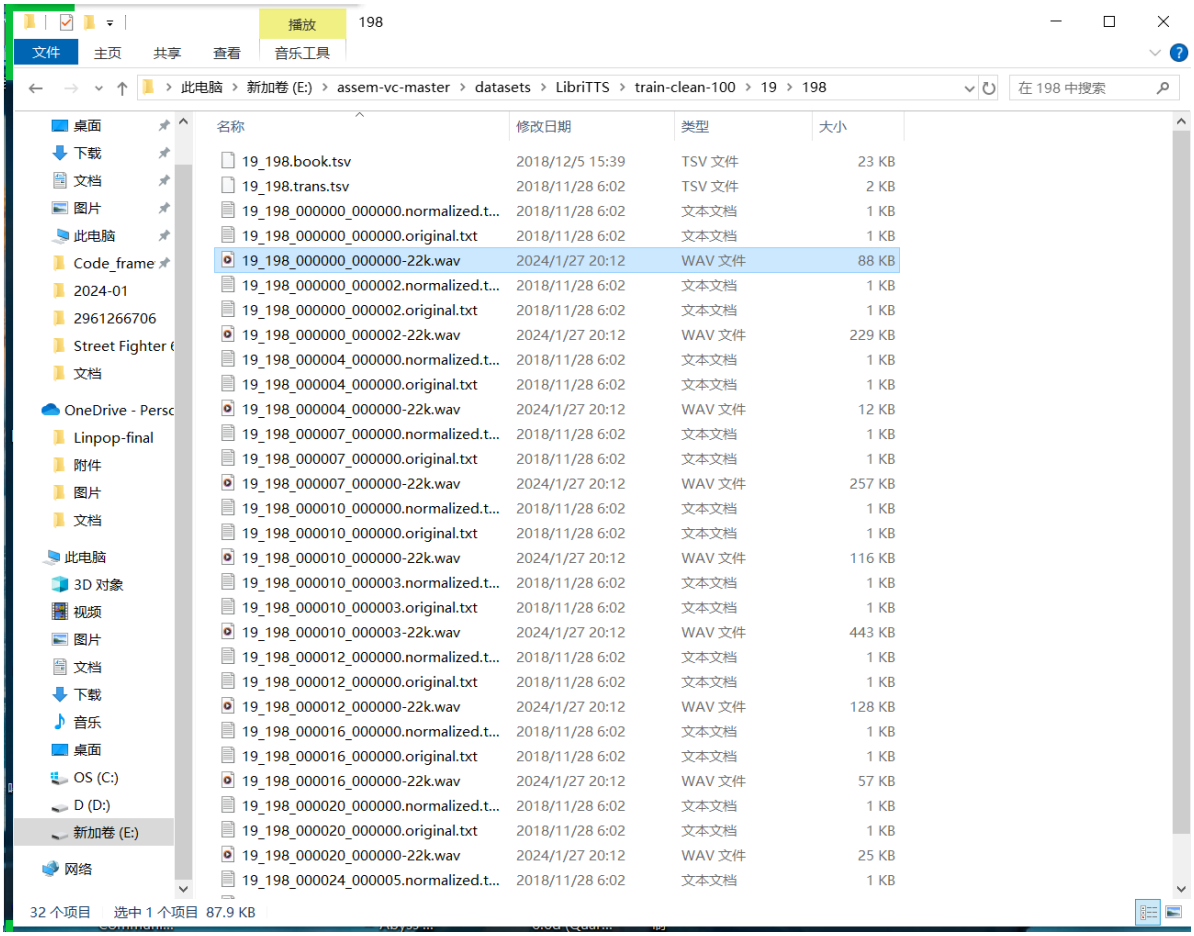
在代码重构方面，talkingface 中的 data 内分别完成了预处理和数据集操作的两个核心部分，并分别对应 dataprocess 和 dataset，具体组织如下：



其中，预处理主要通过 G2PConverter、AudioResampler 和 F0Extractor 三个类完成，该部分截图如下所示：







可以证明其已处理完毕，重采样频率为 22050 后命名格式也与原数据格式变化，修饰后缀 -22k.wav。

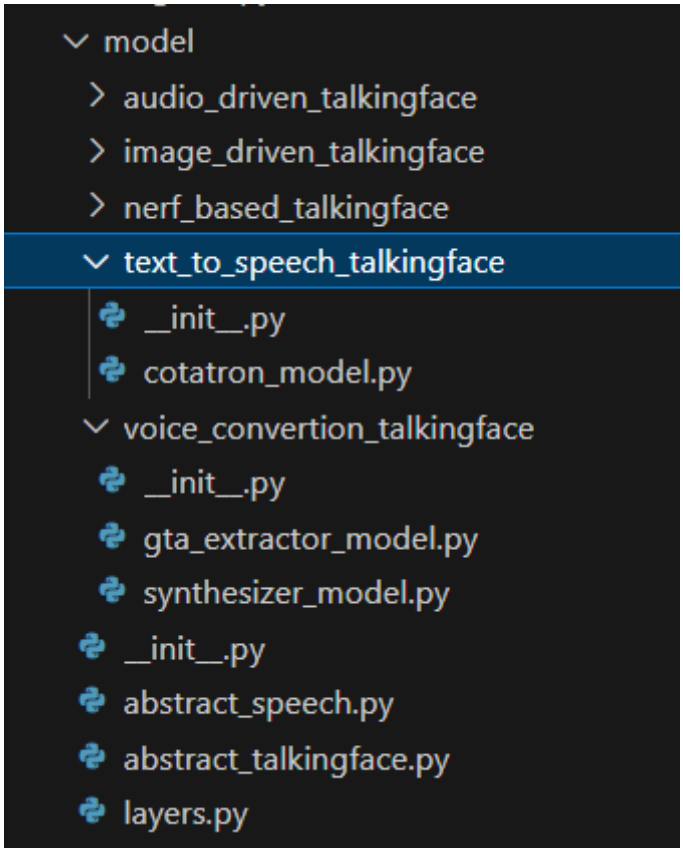
dataset 部分则构建 TextMelDataset 类以继承原框架的 Dataset 抽象类完成相关组织数据集部分的代码整合，具体内容见项目。

3.2 模型与训练

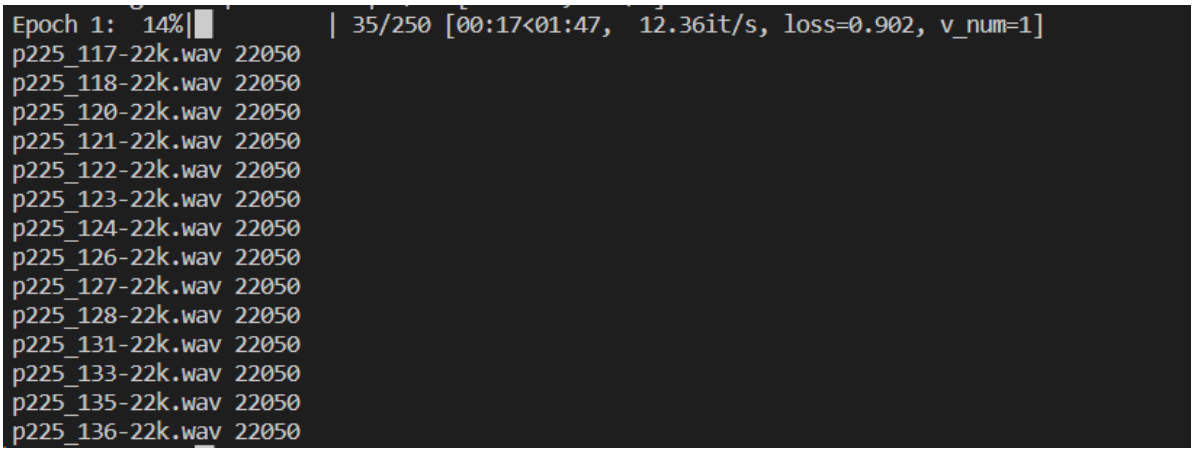
原始项目的模型主要有三个部分，Cotatron、Synthesis、GTA Extractor 分别扮演着不同的功能角色。以下是对每个模块的功能的解析：

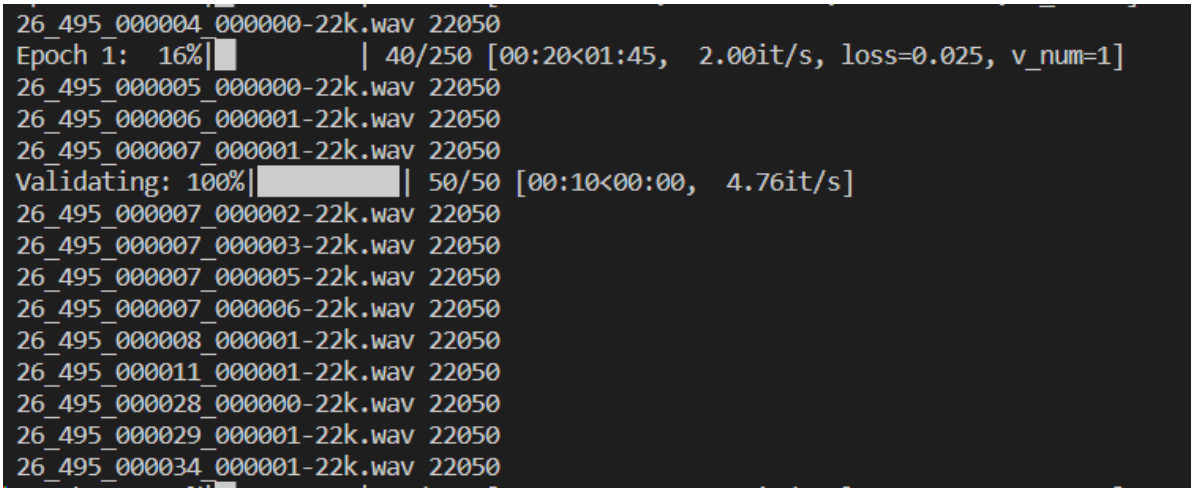
1. **Cotatron**:
    - **功能**：Cotatron用于语音的文本到语音对齐。它负责将输入的文本序列与语音信号进行对齐，以获得语音中每个时间步的语音特征。
    - **角色**：作为语言编码器，Cotatron的输出用于捕获语音的语音特征，用于后续的语音合成过程。
2. **Synthesis**:
    - **功能**：合成模块负责将源说话者的语音特征转换为目标说话者的语音特征，实现说话者之间的语音转换。
    - **角色**：在Assem-VC中，这一阶段的模块主要包括语调编码器、解码器和声码器，负责实际的语音合成过程。
3. **GTA Extractor** (GTA：Ground Truth Alignment)：
    - **功能**：GTA Extractor用于提取Ground Truth Alignment，即从转换后的语音中获取对齐信息。
    - **角色**：在GTA微调阶段，GTA Extractor的输出被用于微调HiFi-GAN声码器，以进一步提高生成语音的质量。

它们被置于 talkingface 框架下 model 中的具体类别，按照分类放置的原则，Cotatron 属于 text to speech 类别，另外两个部分则属于 voice convertion，具体组织如下：



根据上述项目结构，在训练方面，重构部分在 trainer 下的 model_sum_trainer 一方面完成了上述三个模型类的对应训练代码，另一方面整合了大量训练模型的相关方法。其中，三段训练的截图验证如下所示：





```
Epoch 1:   2%|▉                  | 30/250 [00:15<12:30,  0.40it/s, loss=0.205]
Validating: 0it [00:00, ?it/s]
Validating:   0%|                  | 0/50 [00:00<?, ?it/s]
Epoch 1:   6%|▉                  | 35/250 [00:17<10:30,  0.47it/s, loss=0.995]
Epoch 1:   8%|▉                  | 40/250 [00:20<09:50,  0.50it/s, loss=0.810]
Epoch 1:  10%|▉                  | 45/250 [00:22<09:05,  0.54it/s, loss=0.675]
Epoch 1:  12%|▉                  | 50/250 [00:25<08:30,  0.57it/s, loss=0.562]
Epoch 1:  14%|▉                  | 55/250 [00:27<07:55,  0.61it/s, loss=0.467]
Epoch 1:  16%|▉                  | 60/250 [00:30<07:28,  0.64it/s, loss=0.388]
Epoch 1:  18%|▉                  | 65/250 [00:32<07:03,  0.67it/s, loss=0.322]
Epoch 1:  20%|▉                  | 70/250 [00:35<06:43,  0.69it/s, loss=0.267]
Epoch 1:  22%|▉                  | 75/250 [00:37<06:25,  0.72it/s, loss=0.221]
Validating: 100%|▉| 50/50 [00:10<00:00,  5.68it/s]
Epoch 1:  24%|▉                  | 80/250 [00:47<06:05,  0.75it/s, loss=0.182]
Epoch 1:  26%|▉                  | 85/250 [00:50<05:49,  0.77it/s, loss=0.149]
Epoch 1:  28%|▉                  | 90/250 [00:52<05:34,  0.80it/s, loss=0.122]
Epoch 1:  30%|▉                  | 95/250 [00:55<05:21,  0.82it/s, loss=0.099]
Epoch 1:  32%|▉                  | 100/250 [00:57<05:09,  0.84it/s, loss=0.080]
```

然后，通过原项目提供的辅助预训练模型 https://drive.google.com/drive/folders/1all8ObHxsmsFLXBz-y05jMBN4LrpQejm?usp=sharing 和训练保存的 checkpoints 模型以完成推理预测部分，并且在重构代码中能够同样地调用，该部分流程具体截图如下：

```
gdown.download('https://drive.google.com/uc?id=1gt_rLVw83-FMtvTv6PJ5hdqZQRiyaQnh', 'config/global/config.yaml', quiet=False)
gdown.download('https://drive.google.com/uc?id=1kKwchYCAB1A67ELI1LQcpjrYFnSmiPYS', 'config/vc/config.yaml', quiet=False)
gdown.download('https://drive.google.com/uc?id=1dKN4knYOmSBKWZqjP_sX8W_EJSV-7bNP', 'f0s.txt', quiet=False)
gdown.download('https://drive.google.com/uc?id=1k2uJBh3FEK38pCdH38-MaXpgJFb5sEk6', 'pretrained_decoder.ckpt', quiet=False)
gdown.download('https://drive.google.com/uc?id=1Q1XCoY_SKno34dLLm1zNu6PtKuqumXYi', 'hifigan_vctk_g_2600000.ckpt', quiet=False)
```

```
Downloading...
From: https://drive.google.com/uc?id=1gt_rLVw83-FMtvTv6PJ5hdqZQRiyaQnh
To: /content/assem-vc/config/global/config.yaml
100%|██████████| 3.06k/3.06k [00:00<00:00, 1.04MB/s]
Downloading...
From: https://drive.google.com/uc?id=1kKwchYCAB1A67ELI1LQcpjrYFnSmiPYS
To: /content/assem-vc/config/vc/config.yaml
100%|██████████| 338/338 [00:00<00:00, 94.0kB/s]
Downloading...
From: https://drive.google.com/uc?id=1dKN4knYOmSBKWZqjP_sX8W_EJSV-7bNP
To: /content/assem-vc/f0s.txt
100%|██████████| 9.81k/9.81k [00:00<00:00, 2.32MB/s]
Downloading...
From: https://drive.google.com/uc?id=1k2uJBh3FEK38pCdH38-MaXpgJFb5sEk6
To: /content/assem-vc/pretrained_decoder.ckpt
168MB [00:00, 183MB/s]
Downloading...
From: https://drive.google.com/uc?id=1Q1XCoY_SKno34dLLm1zNu6PtKuqumXYi
To: /content/assem-vc/hifigan_vctk_g_2600000.ckpt
55.8MB [00:00, 145MB/s]
'hifigan_vctk_g_2600000.ckpt'
```

以上是获取所需的预训练模型。

```
audio_path, text, _ = targetloader.meta[file_idx]
x = targetloader.__getitem__(file_idx)

batch = text_mel_collate([x])

print(text)
```

```
{Y AO R} {T AY M} {IH Z} {L IH M AH T AH D}, {S OW} {D OW N T} {W EY S T} {IH T} {L IH V IH NG} {S AH M W AH N} {EH L S IH Z} {L AY F}. {D OW N T} {
```

```
x, sr = librosa.load(os.path.join('datasets/inference_source', audio_path))
ipd.Audio(x, rate=hp.audio.sampling_rate)
```

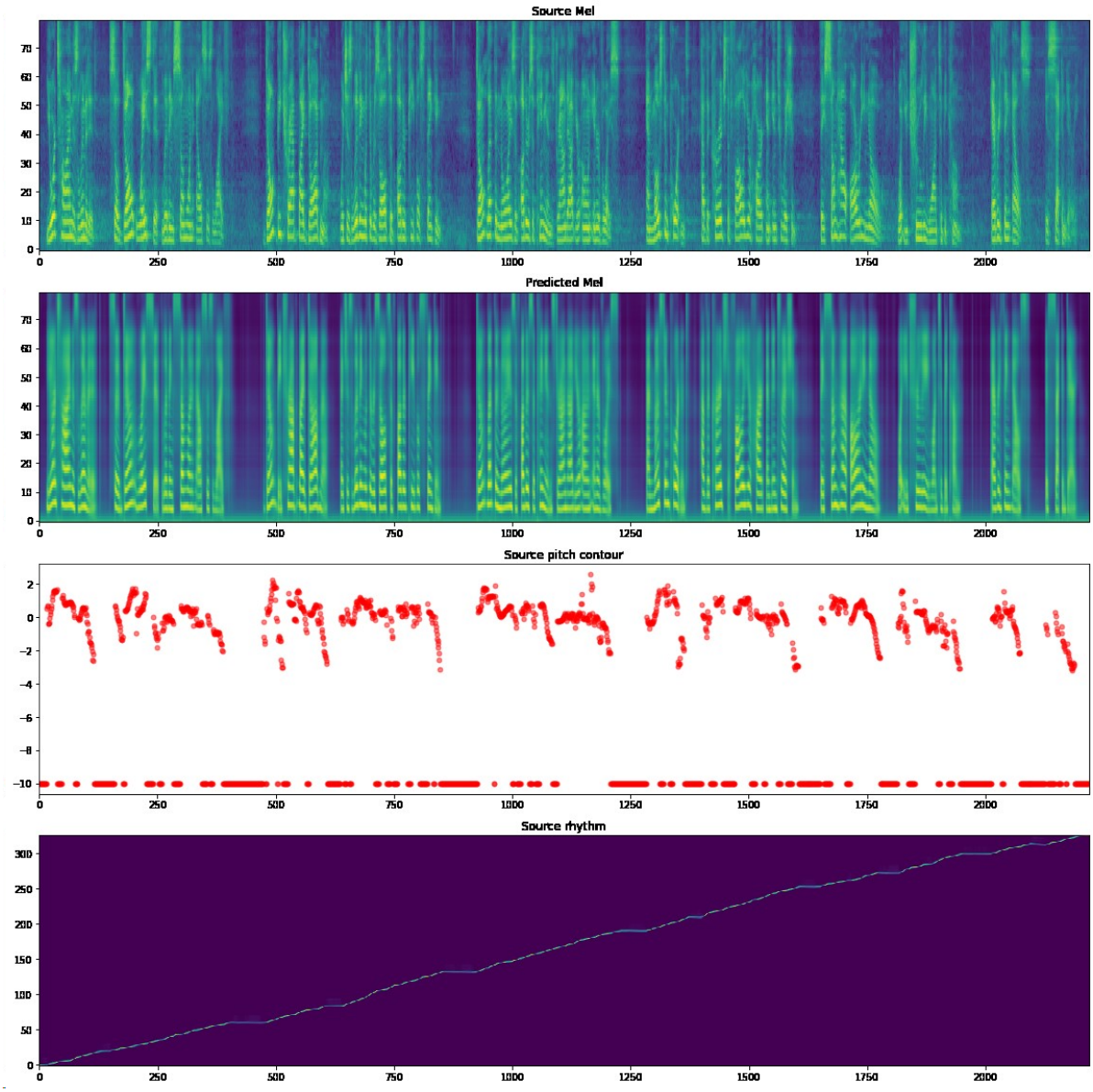
0:00 / 0:25

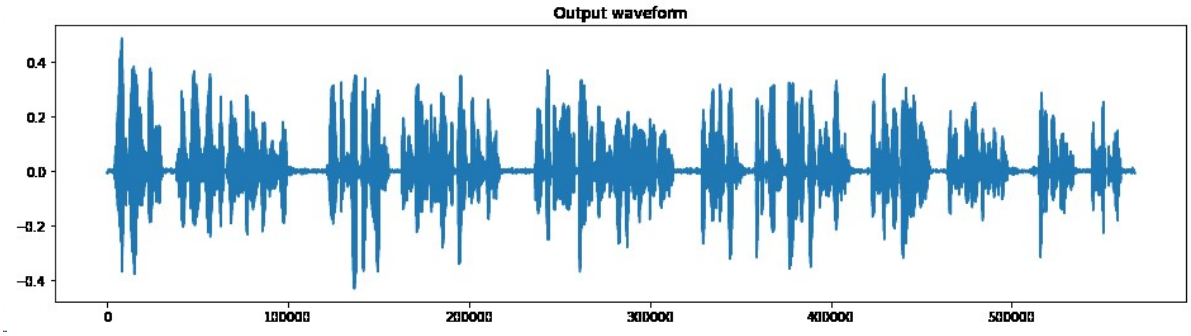
```
file_idx = random.randrange(len(dataloader))
audio_path, _, _ = dataloader.meta[file_idx]
x = dataloader.__getitem__(file_idx)
target_batch = text_mel_collate([x])
x, sr = librosa.load(os.path.join(target_root, audio_path))
ipd.Audio(x, rate=hp.audio.sampling_rate)
```

0:00 / 0:03

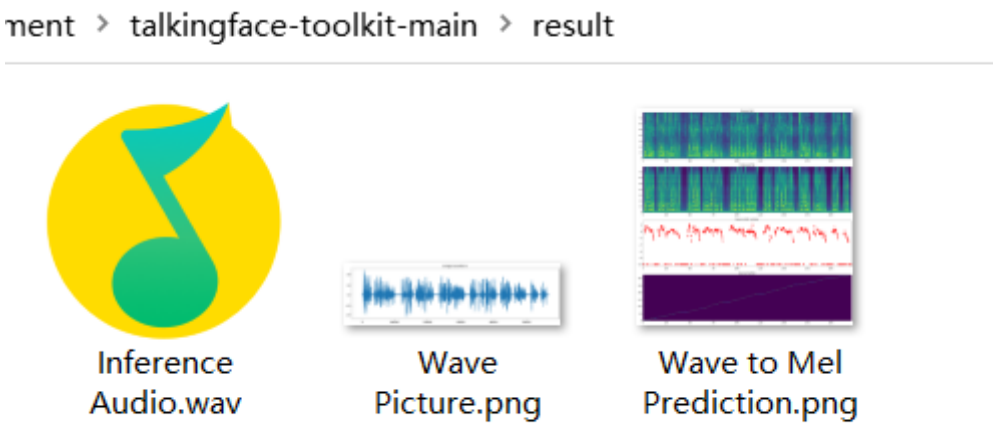
```
ipd.Audio(audio, rate=hp.audio.sampling_rate)
```

0:00 / 0:25





以上为预测音频的生成，原始 Mel 频谱与预测部分的可视化对比，以及输出波形图，具体的文件均保存在 result 文件夹下，证明截图如下：



综上部分，在 properties 中重新整合了 .yaml 格式的三个 config 配置文件，控制训练等过程的参数等设置。此外，以 quick_start 模块和 run_talkingface.py 相契合的数据结构形式把以上内容组织起来，能够以 python 命令形式走完从数据处理组织到模型训练预测等步骤的全流程。

4 额外优化与改进

在基本要求全部完成后，我们对重构后的项目部分涉及模型训练的部分进行了额外的改进和优化，主要包括两个部分：Attention 类的注意力机制优化、

4.1 Attention 类的注意力机制优化

关于对 Attention 类的改进，这个注意力机制与训练过程相关，其使用了传统的 masked 式注意力方法，对此有以下优化，以增强其在具体情境下的上下文联系能力。

- 创建一个新的类 `MultiHeadAttention`，它包含多个 `Attention` 头。
- `num_heads` 参数决定了使用的注意力头的数量。

```
class MultiHeadAttention(nn.Module):
    def __init__(self, num_heads, attn_rnn_dim, attn_dim, static_channels, static_kernel_size,
                 dynamic_channels, dynamic_kernel_size, causal_n, causal_alpha, causal_beta, dropout_prob):
        super(MultiHeadAttention, self).__init__()
        self.num_heads = num_heads

        # Define multiple attention heads
        self.attention_heads = nn.ModuleList([
```

```
            Attention(attn_rnn_dim, attn_dim, static_channels,
static_kernel_size,
                      dynamic_channels, dynamic_kernel_size, causal_n,
causal_alpha, causal_beta)
            for _ in range(num_heads)
        ])

        # Dropout layer
        self.dropout = nn.Dropout(p=dropout_prob)
```

- 在 `forward` 方法中，对每个注意力头调用其 `forward` 方法。
- 将所有注意力头的输出沿着特征维度进行堆叠。
- 使用平均池化 `torch.mean` 方法对多头的输出进行池化。
- 应用 dropout，以防止过拟合。

```
    def forward(self, attn_hidden, memory, prev_attn, mask):
        # Apply each attention head separately
        attention_heads_output = [head(attn_hidden, memory, prev_attn, mask)
[0] for head in self.attention_heads]

        # Concatenate attention head outputs along the feature dimension
        combined_output = torch.stack(attention_heads_output, dim=-1)
        combined_output = torch.mean(combined_output, dim=-1)  # Average
pooling

        # Apply dropout
        combined_output = self.dropout(combined_output)

        return combined_output, None  # Return None for attn_weights for
simplicity
```

这种改进的主要思想是将多个 Attention 头组合在一起，以捕获更丰富的信息，并通过平均池化对这些头的输出进行整合，并且添加了 dropout 来防止过拟合，综合来说增强了该类的实现功能。

4.2 引导注意力丧失的改进

原始代码中，使用了引导注意力丧失来辅助加速 Cotatron 的训练过程以达到更快地收敛，我们对这一机制进行了优化，以下是两个主要的思路方向。

- **性能优化**： 使用 PyTorch 的向量化操作和内置函数，以提高计算效率。
- **数值稳定性**： 使用 PyTorch 提供的稳定数学操作，减少数值稳定性问题。

```
import torch

class GuidedAttentionLoss(torch.nn.Module):
    def __init__(self, guided_att_steps, guided_att_variance,
guided_att_gamma):
        self._guided_att_steps = guided_att_steps
        self._guided_att_variance = guided_att_variance
        self._guided_att_gamma = guided_att_gamma

    def set_guided_att_steps(self, guided_att_steps):
        self._guided_att_steps = guided_att_steps

```

```
    def set_guided_att_variance(self, guided_att_variance):
        self._guided_att_variance = guided_att_variance

    def set_guided_att_gamma(self, guided_att_gamma):
        self._guided_att_gamma = guided_att_gamma

    def forward(self, alignments, input_lengths, target_lengths,
global_step):
        if self._guided_att_steps < global_step:
            return 0

        self._guided_att_variance = self._guided_att_gamma ** global_step

        weights = self._compute_guided_attention_weights(
            alignments, input_lengths, target_lengths)

        loss = torch.sum(weights * alignments) /
target_lengths.float().sum()

        return loss

    def _compute_guided_attention_weights(self, alignments, input_lengths,
target_lengths):
        weights = torch.zeros_like(alignments)
        for i, (f, l) in enumerate(zip(target_lengths, input_lengths)):
            grid_f, grid_l = torch.meshgrid(
                torch.arange(f, dtype=torch.float, device=f.device),
                torch.arange(l, dtype=torch.float, device=l.device))
            weights[i, :f, :l] = 1 - torch.exp(
                -((grid_l / l - grid_f / f) ** 2) / (2 *
self._guided_att_variance ** 2))
        return weights
```

通过使用向量化操作和 PyTorch 内置函数，我们观察到代码的计算效率得到了提高。在相同的训练步骤下，优化后的代码在数值稳定性方面也表现更好。

5 成员分工

本小组成员包括：吴隽恺（组长，学号 1120213607）、刘吉昊（组员，学号 1120212843）、王治桐（组员，学号 1120212171）。具体分工如下：

- 吴隽恺：主要负责训练和预测相关的代码重构，即 train 目录下的内容，对上述三个模型的训练、推理预测、统合模型与相关机制引入等部分进行重构和整合实现；设计并完成了上述两项对原重构结果的额外优化与改进。
- 刘吉昊：主要负责数据相关的代码重构，即 data 目录下的内容，对上述数据集预处理、设计字典结构、组织编码等部分进行重构与整合实现。
- 王治桐：主要负责模型相关的代码重构，即 model 目录下的内容，对上述三个模型的架构搭建、抽象类继承和具体的内容功能设计完成了重构与整合。
- 注：未说明的其他杂项部分由三名小组成员共同合作完成。

6 附加说明

为解决 .md 文件出现非本地无法正常显示图片的情况，我们备份了一份同样的 .pdf 文件以防该情况发生。此外，在 Github 上传该项目作业文件中遇到一些传输文件大小受到限制的问题，故我们删去了部分阻碍这一过程的过大的数据集以及保存的模型等内容，在此前的报告内容中也已有截图可证明完成此流程，特此附加说明。