StarGANv2 项目说明

项目地址：https://github.com/thsno02/StarGANv2-VC

模型简介

StarGANv2-VC 是一个无监督的、使用非平行语料的、多对多的语音转化模型，该模型使用 StarGAN v2 生成对抗网络。

论文地址：https://arxiv.org/abs/2107.10394

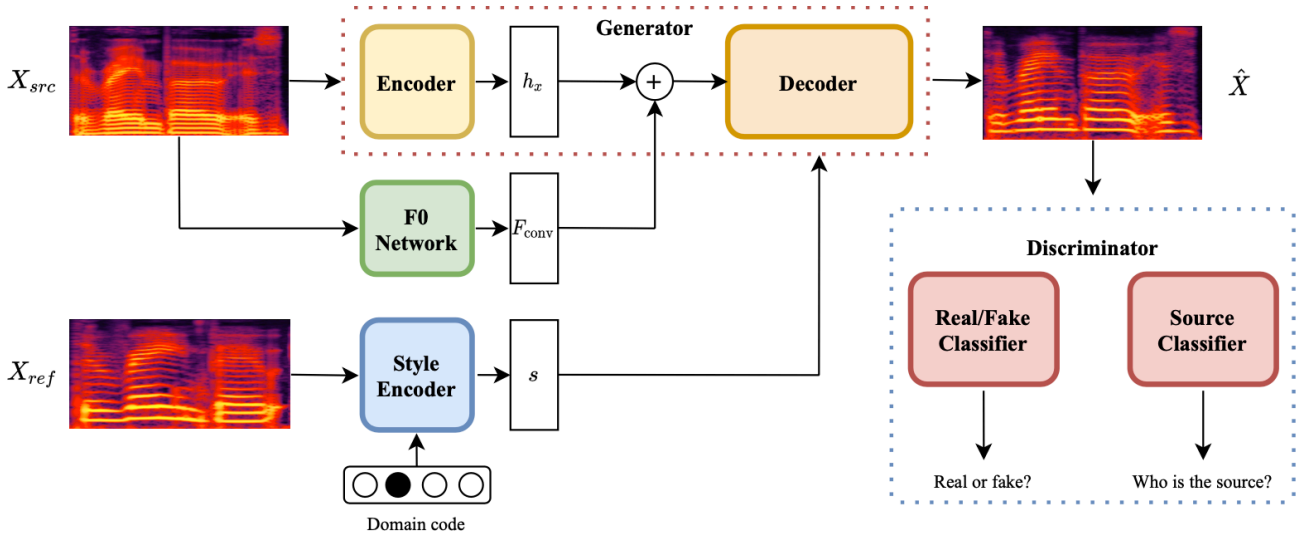
GitHub地址：https://github.com/yl4579/StarGANv2-VC

论文示例：https://starganv2-vc.github.io/

模型特点：

- 模型的泛化性较强。
    - 虽然目前仅使用了 10 个人，每人 20 分钟的语音样本进行训练，但是对于某些训练过的发音者，任意人的输入都能得到一个可接受的转化效果。可接受指，在符合一些规则下，能够保留大量的语言信息，且转化声音带有发音者的语音语调。一些规则是，语音稍慢，不要带有太强的语音语调。
    - 虽然模型中的两个预训练模型皆使用的英文语音数据集进行训练，但是也能适用于中文的场景。
    - 模型可以对方言进行转化，而不仅局限于普通话。经测试，使用四川话进行转化能取得可接受的效果。
- 模型可以将无起伏的语音转化为有语音语调的语音。
- 模型的转化速度很快，可以实现比实时更快（faster-than-real-time），因此，实时转化是可以实现的。

模型架构：



模型训练：

- 训练时间：200 小时，5 路 3080。
- 数据：
    - 发音者：10 个人。
    - 音频时长：每人 20 分钟的音频，训练使用的音频是 2 秒的音频片段。

    - 数据筛选：使用的音频片段的经过人工筛选，确保每一个片段至少有 4 个汉字。

实训评估：

- 转化速度很快，实施实时转化没有任何模型方面的阻碍。目前因为展示方式，呈现的是「非实时的离线」转化。
- 模型的泛化性不错，在中文上的表现可接受，方言也能转化。
- 可能因为数据集的不规范，目前只有两位发音者能有比较好的转化效果，其它人的效果难以接受。

文件说明

`./Configs`：模型的参数设置。

`./Data`：训练相关的数据。

- 每一个人名文件夹对应一个发音者的音频数据；`raw` 文件夹中储存从发音者视频 `./Video` 中提取到的原音频。
- 发音者文件夹中存储的都是时长为 2s 的音频片段，该片段由 `raw` 中的长音频文件剪切而成。
- 筛选后的音频数据列表储存在 `./Data/train_list.txt` 和 `./Data/val_list.txt` 便于调用。

> 不是所有数据都进行了筛选，每人只筛选了二十四分钟可用的数据。

`./Models`：模型的 checkpoints。

`./Pred`：需要进行预测/转化的数据。

- `./Pred/A` 是待转化的数据集合。
- `./Pred/A/name` 是待转化数据集合中某人的语音数据。

`./Output`：使用批量转化脚本 `offline_infer.py` 转换 `./Pred/A` 中得到的数据。

- 每一个人名文件夹对应一个发音者的音频数据。
- 对于同一个预测数据，使用了两种转化方法「style conversion」和「mapping conversion」，使用后缀「_sty_」和「_map_」进行区分。

`./Utils`：英文的预训练模型。

- `./Utils/ASR`：ASR = automatic speech recognition
    - 作用：进行语音类的文字识别。
- `./Utils/JDC`：JDC = joint detection and classification F0 extraction network
    - 作用：获取语音相关的 F0 频率信息。

`./Video`：发音者的音频文件，用于提取音频

`./Vocoder`：发声器

- 作用：自动生成声音。
- 下载地址：https://drive.google.com/file/d/1q8oSAzwkqi99oOGXDZyLypCiz0Qzn3Ab/view。

预处理使用的文件：

- `prepreocess.ipynb`：按需要运行相关的代码块。

训练使用的文件：

- `losses.py`：损失函数相关；
- `meldataset.py`：梅尔图谱相关；
- `models.py`：模型建构相关；
- `optimizers.py`：优化算子相关；
- `train.py`：模型训练的入口函数；
- `trainer.py`：训练相关的类；
- `transforms.py`：数据转化相关。

推理使用的文件：

- `offline_infer.py`：批量推理
  - 输入：`./Pred` 里的文件；
  - 输出：`./Output` 里的文件；
  - 耗时：5±2 mins。
- `real_time_infer.py`：单条推理
  - 输入：实时的整条语音；
  - 输出：转化后的整条语音；
  - 耗时：初次使用时，10s 音频耗时1.5±0.2s；复用时，10s 音频耗时0.5±0.2s。
- `realtime_infer.py`：实时推理，论文作者写的
  - 输入：实时的语音流；
  - 输出：转化后的语音流；
  - 耗时：未测试。

其它文件：

- `realtime_infer.ipynb`：调试实时推理的文件。
- `sounddevice.ipynb`：调试 soundevice 和语音流的文件。
- `device_test.py`：测试终端语音输入和输出设别的文件。
- `requirements.txt`：本项目的依赖文件。
- `setup.sh`：伪一键部署文件。

细节说明

所有音频数据的采样率（sample rate）皆需要为 24k，否则模型没法得到好的结果，因为预训练模型是基于 24k 采样率做的。

- 环境部署:
  - 需先安装 Anaconda，方便进行环境管理；
  - Windows，在 Git BASH 中运行 `bash setup.sh`； macOS， 在 Terminal 中运行 `sh setup.sh`。
- 数据处理：
  - 将需提取音频的视频文件放入 `./Video` 里，命名方法为 `./姓_名/姓_名_n.mp4`，其中姓和名的首字母大写，比如 `Li_Xiaohua, Cheng_Long`；
  - 按需求运行 `prepreocess.ipynb` 中的代码块进行数据数据。
- 训练模型：

    - 在 `./Config/config.yml` 中修改相应的参数。目前的版本是多 GPU 训练的版本，如果需要进行单 GPU 训练，参考原论文作者的代码仓库；
    - 开始训练，在 Terminal 里执行：

      ```
      python train.py --config_path ./Configs/config.yml
      ```

- 模型推理：
    - 使用 `offline_infer.py` 将对 `./Pred` 里的文件进行推理，每一个文件会根据 style 和 mapping 转化器进行两次转化，以期比较哪种效果更好。转化后的文件放在 `./Output/speaker_name` 文件夹中，每一个文件都将转化为所有的发音者语音。
    - 使用 `real_time_infer.py` 进行实时语音录入，待录入结束后对整条语音进行转化，可根据相关的提示选取语音输入设备、语音输出设备、发音者。