|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **TTS语音合成**软件过程文档 | | | | |
| 内部编号： | | | 文档日期：**2023.1.1** | | 共 **8** 页 |
| 作者: 刘宜松 | | | 版本号： | | 上一版本：**\** |
| 文档类型： | | | 工作起始时间： | | |
| 主要参加人员： | | | | | |
| 审定： | | 日期： | | 文件编号： | |
| 参考文档**(**内部**)**：  参考文档**(**其他**)**： | | | | | |
| 内容摘要：  语音合成（Text To Speech，TTS）技术将文本转化为声音，目前广泛应用于语音助手、智能音箱、地图导航等场景。TTS的实现涉及到语言学、语音学的诸多复杂知识，因合成技术的区别，不同的TTS系统在准确性、自然度、清晰度、还原度等方面也有着不一样的表现，如何从多维度综合评价TTS系统质量成了TTS测试人员的一大挑战。针对TTS前端、后端的存在的问题，选取TTS评测指标，制定各指标评测方法，形成了一套系统的TTS评测方案。 | | | | | |

目 录

目 录 [2](#_bookmark1)

1. 系统概述 [3](#_bookmark2)

2. 软件需求分析 [3](#_bookmark3)

2.1 灵活配置 [3](#_bookmark4)

2.2 控制量获取方式 4

2.3 时间片化 4

2.4 数据流设计要求-序列化 4

2.5 时间片化 4

2.6 序列化要求 4

3. 概要方案 5

3.1 配置文件 5

3.2 PV 数据获取方式 5

3.3 PulseID 6

3.4 Flatbuffer 序列化 6

3.5 方案的重点与难点 6

4. 软件运行环境 6

5. 软件设计 7

5.1 Metadata 7

5.2 总体流程设计 8

5.3 配置文件 8

5.4 PV\_Scanner 线程线程设计 9

5.5 主线程设计 9

5.6 定时器 9

5.7 提交线程 9

6. 代码实现 10

6.1 程序包结构 10

6.2 配置文件解析的实现 10

6.3 主线程实现 11

6.4 定时器实现 11

6.5 PV\_Scanner 实现 12

7. 软件操作界面 12

8. 软件测试 12

9. 上线运行 13

1. 数据准备
   1. 数据格式要求

可以说整个项目大部分时间花费在整理数据集上，根据我自己的经验，数据集的语音长度在2秒-10秒之间效果最好，数量大约在2000条左右（为了涵盖尽可能多的汉字发音）。需要注意一点，不管拆包的原语音采样率如何，都要统一重采样到22050 hz，这是Tacotron2训练模型的要求。

1.2 FFmpeg介绍

为了将语音采样率设置为22050hz以满足Tacotron2的训练要求，我需要介绍一下FFmpeg。

FFmpeg是一套可以用来记录、转换数字音频、视频，并能将其转化为流的开源计算机程序。采用LGPL或GPL许可证。它提供了录制、转换以及流化音视频的完整解决方案。它包含了非常先进的音频/视频编解码库libavcodec，为了保证高可移植性和编解码质量，libavcodec里很多code都是从头开发的。

采样率的处理脚本

import os  
import subprocess  
  
input\_path="/home/ubuntu/Desktop/testing"  
output\_path="/home/ubuntu/Desktop/testing\_new"  
for file in os.listdir(input\_path):  
 file1=input\_path+'/'+file  
 file2=output\_path+'/'+file  
 cmd="ffmpeg -I "+file1+" -ar 22050 "+file2  
 subprocess.call(cmd,shell=True)

* 1. 文本格式准备

需求分析：以中文标准女声音库（10000句)为例子

链接：https://aistudio.baidu.com/aistudio/datasetdetail/117129

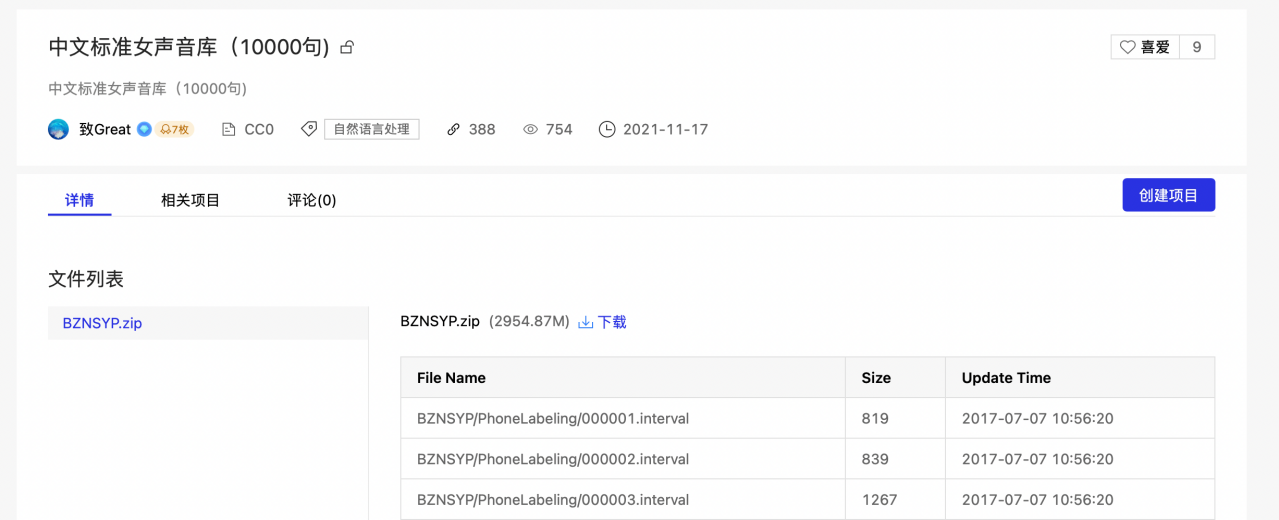
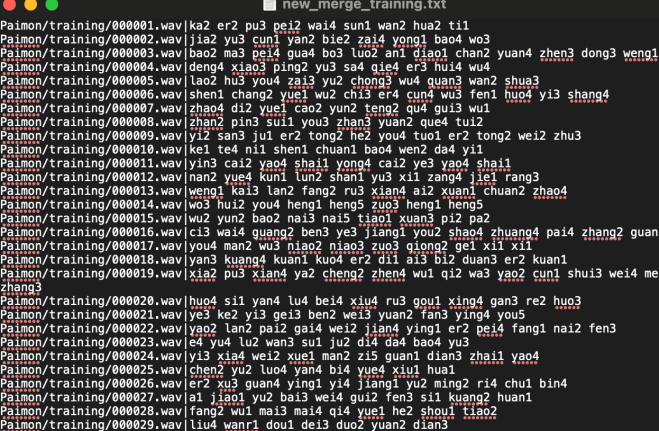
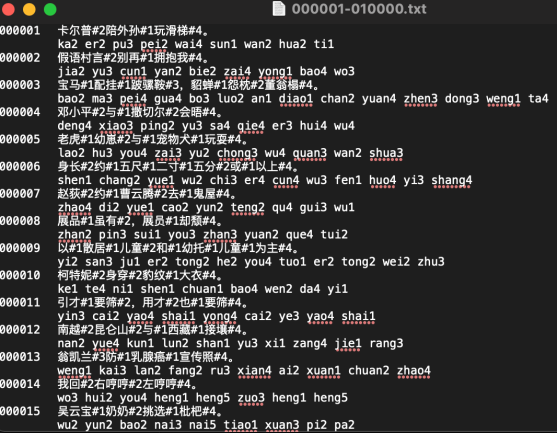


图1 语料库来源



左图为原始文本，右图为目标文本

左图的特征：序号+中文(含有#符号)+换行+拼音

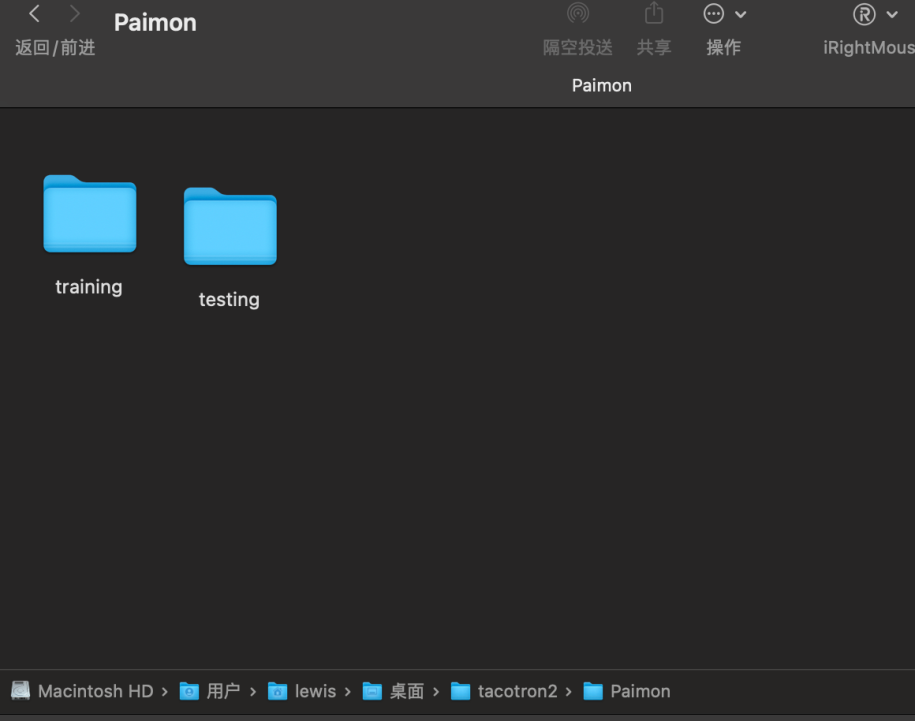
右图的特征：路径.序号.wav+|符号+拼音

处理方案：VSCode+正则表达式

1.3 语料库使用

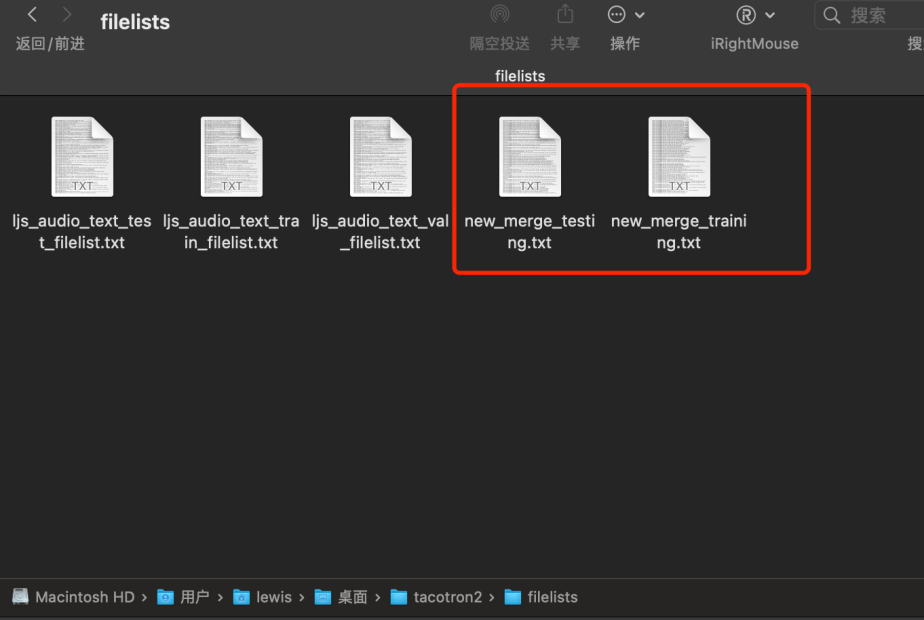
将处理好的语料库按照3：7的比例划分数据集为训练集和测试集。

存放路径：/Users/lewis/Desktop/tacotron2/Paimon



文本也按照对应的序号进行划分

存放路径：/Users/lewis/Desktop/tacotron2/filelists



2 硬件/软件版本要求

模型训练

硬件：独立显卡，显存>1G

软件：

1. tensorflow-gpu>2.0
2. matplotlib==2.1.0
3. numpy==1.13.3
4. inflect==0.2.5
5. librosa==0.6.0
6. scipy==1.0.0
7. Unidecode==1.0.22
8. Pillow

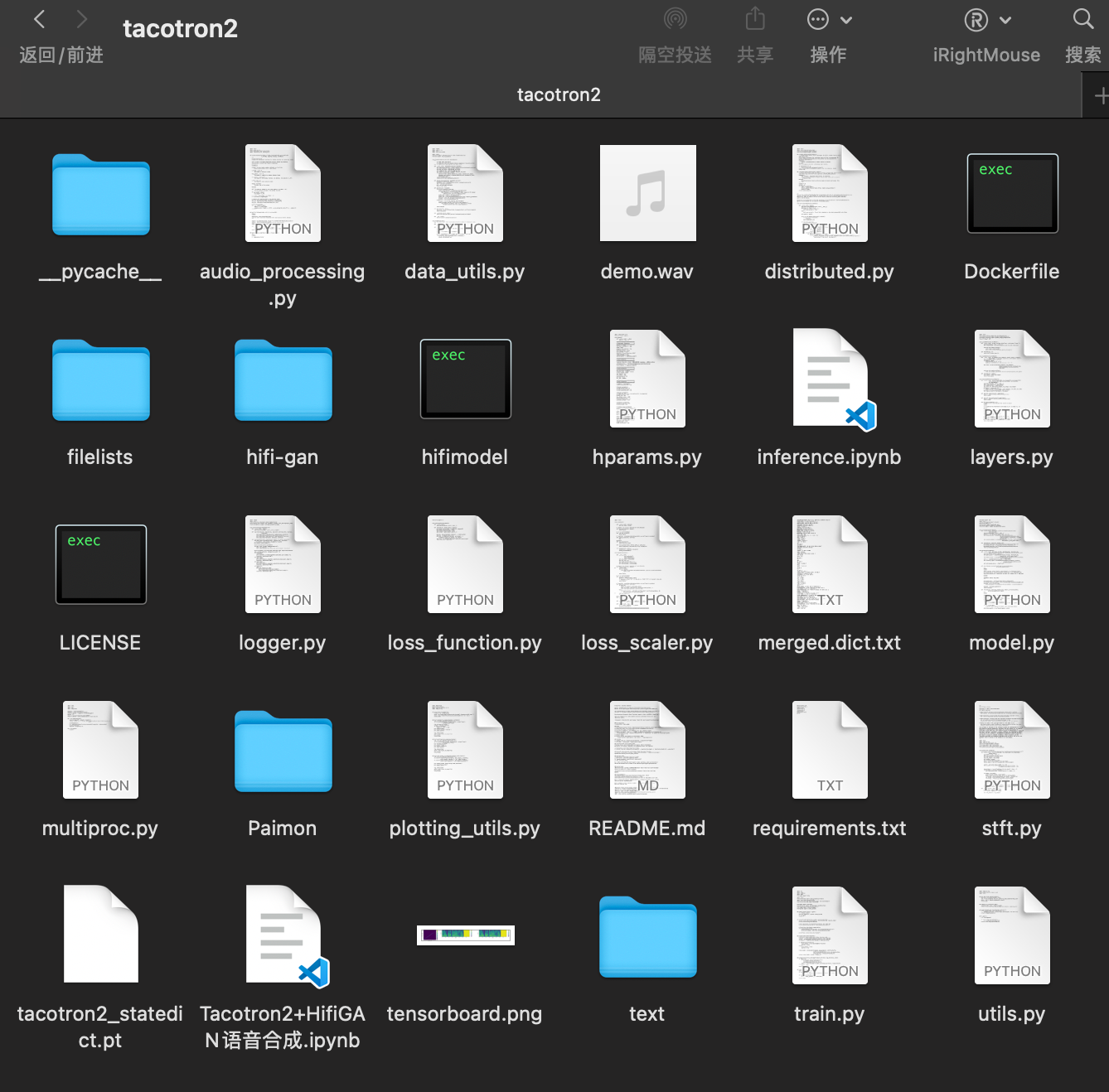
模型推导

硬件：独立显卡，显存>1G（可只用CPU推导）

软件：

1. torch==1.4.0
2. numpy==1.17.4
3. librosa==0.7.2
4. scipy==1.4.1
5. tensorboard==2.0
6. soundfile==0.10.3.post1
7. matplotlib==3.1.3

项目结构



tacotron2：主文件，来源是github源文件

Tacotron2+HifiGAN语音合成.ipynb：项目主文件，启动文件

hifi-gan：声码器文件夹，来源是github源文件

Paimon：语音库文件夹

filelists：文本文件夹

## 项目使用

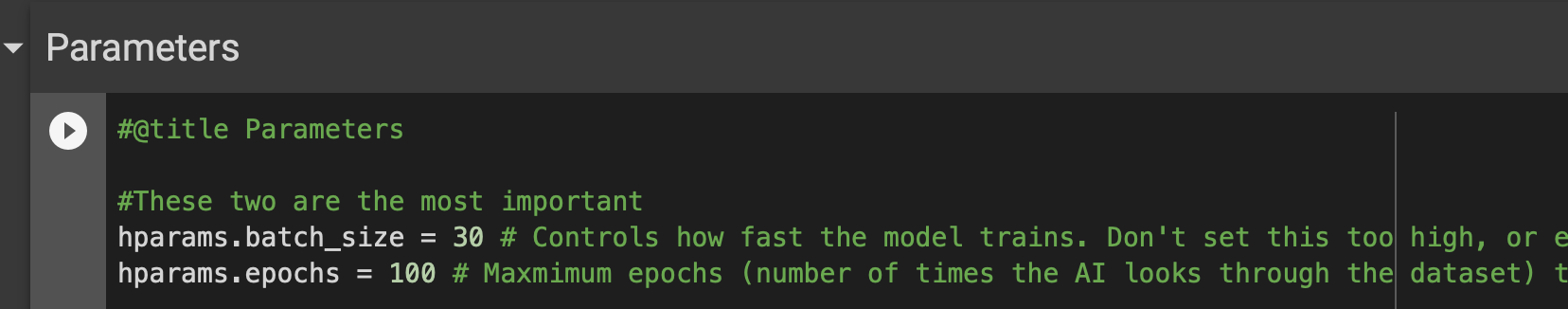
### 重要参数：



model\_filename=模型名称

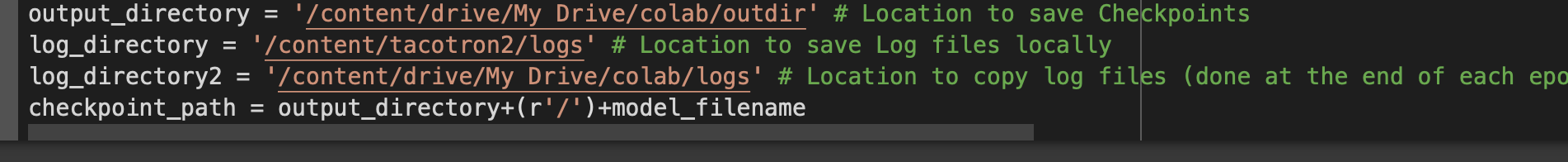
hparams.training\_files = 训练集文本的存放路径

hparams.validation\_files=测试集文本的存放路径



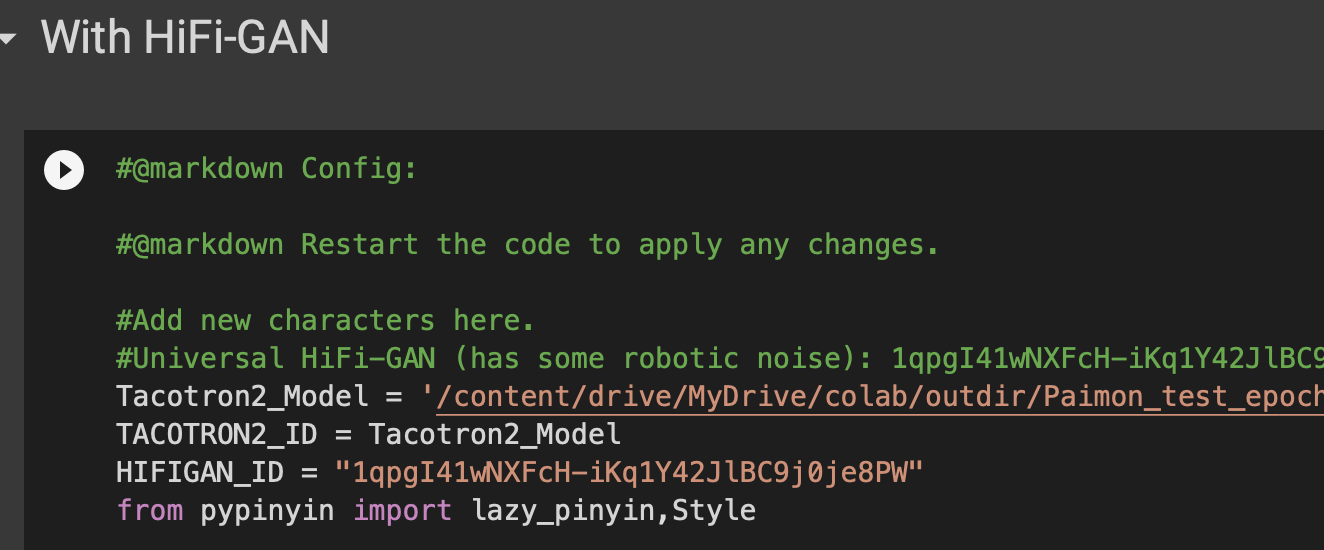
Hparams.batch\_size=batch大小，训练速度

Hparams.epochs=epoch大小，影响训练速度



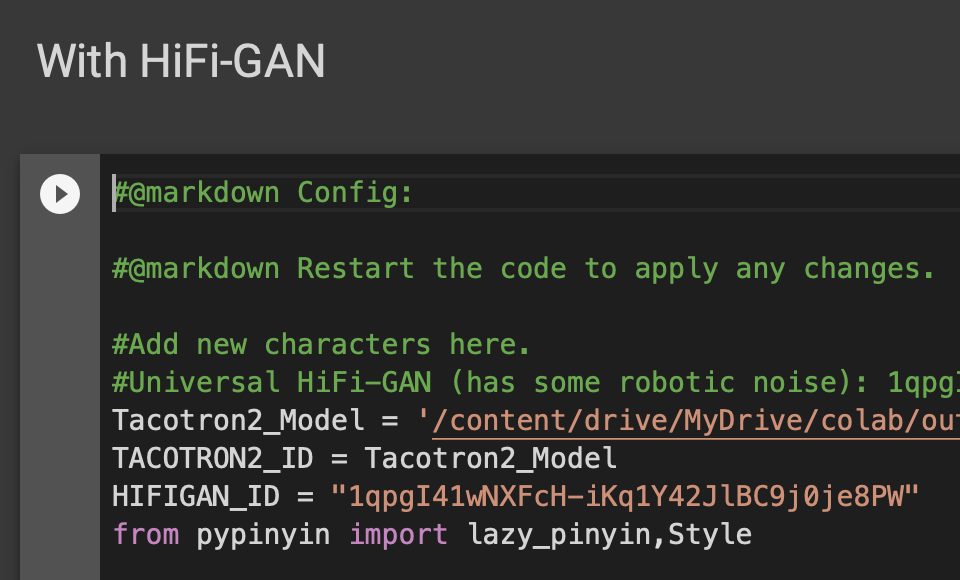
output\_director=模型的输出路径

log\_directory=模型的日志路径



Tacotron2\_Model=模型的名称，需要和之前的model\_filename对应

### 项目启动



1. Tacotron2介绍

简单讲一讲Tacotron2，它是由google推出的从文本中合成语音的神经网络结构，也就是一个语音合成（Text To Speech，TTS）框架，可以实现端到端的语音合成。Tacotron2与其前代Tacotron类似，比较重要的一个区别是在编码器模块中引入了一个双向LSTM层和卷积层，相比原来的CBHG堆叠结构和GRU循环层更为简洁。

模型主要由两部分组成：

声谱预测网络：特征预测网络，包含一个编码器和一个引入注意力机制（attention）的解码器，作用是将输入字符序列预测为梅尔频谱的帧序列。

声码器（vocoder）：将预测的梅尔频谱帧序列转换产生时域波形样本，算是WaveNet的修订版。

原项目中的声码器我们暂时不用（上面地址提供的Tacotron 2就是没有wavenet的版本），因为有更好的工具HiFi-GAN。

代码实现详解有很多博客可以参考（(16条消息) Tacotron2 论文 + 代码详解\_HJ\_彼岸的博客-CSDN博客\_tacotron2），这里只要知道我们是用Tacotron2生成梅尔频谱，在此基础上结合我们输入的字符序列（也就是对应的拼音文本）训练模型。

特别注意一点：Tacotron 2是基于tensorflow1.5版本运行的，如果是自己电脑上配置环境的话，务必将python版本降到3.7以下！否则将会无法安装tensorflow1.5，除了tensorflow有硬性版本要求之外，其他依赖都可以安装最新版本

3 HiFi-GAN简介

简单说下，声码器的作用就是将梅尔频谱转换成语音信号，和上面是对应的。

为什么我们没有用上面Tacotron2的声码器呢，主要原因就是现在有很多更优秀的声码器供我们选择。

早期比较有名的声码器WaveNet，它是一种自回归卷积神经网络，合成的效果非常好可以说和人类发声非常相似，但有个致命的缺点——合成速度太慢。直到2020年项目作者开发了这套基于GAN（生成式对抗网络）的神经网络声码器，从作者的论文里可以找到，HiFi-GAN在GPU上可以以比实时速度快167.9倍的速度生成22.05 kHz的语音，在CPU上可以以比自回归模型快13.4倍的速度生成语音，这就是它的牛逼之处。

HiFi-GAN主要有一个生成器和两个判别器，具体结构就不说了，知道一下生成器和两个判别器是通过对抗学习的方法训练的，新增加了两个损失函数来提高训练的稳定性和提高模型的性能。有能力的小伙伴可以看原论文（HiFi-GAN: Generative Adversarial Networks for Efficient and High Fidelity Speech Synthesis）了解详情。

需要注意一下作者使用VCTK数据集进行实验，测试了3个模型（V1、V2和V3），简单来说V1是最优模型，作者发布的预训练模型以及相应的配置文件都是以V1模型为基础的。我在这篇博客使用的HiFi-GAN模型g\_02500000就是作者的预训练模型，配置文件为config.json。