**智能语音识别大作业**

22009200601汤栋文

2024年12月12日

**目录**

[摘要 2](#_Toc184934104)

[1. 相关背景 2](#_Toc184934105)

[2. 模型介绍 2](#_Toc184934106)

[**2.1 基本结构 2**](#_Toc184934107)

[**2.2 设计核心 3**](#_Toc184934108)

[3. 实验结果 3](#_Toc184934109)

[**3.1 实验环境 3**](#_Toc184934110)

[**3.2 超参数 4**](#_Toc184934111)

[**3.3 实验结果 4**](#_Toc184934112)

[**3.3.1连读与弱读细节 4**](#_Toc184934113)

[**3.3.2轻微失真问题改善 4**](#_Toc184934114)

[**3.3.3生成速度提高 4**](#_Toc184934115)

[**3.3.4频谱图可视化 5**](#_Toc184934116)

[**3.4 不足与改进 5**](#_Toc184934117)

[4. 结论 5](#_Toc184934118)

[引用 5](#_Toc184934119)

# 摘要

Tacotron是一种端到端的文本到语音的生成模型，采用Encoder-Attention-Decoder框架和RNN结构。但现在看来原始的Tacotron模型存在一些问题，特别是RNN和自回归的串行机制使得训练和推理过程效率较低。Parallel Tacotron 2 [1]通过非自回归生成机制显著提高了生成速度和效率，并且通过可微分的持续时间建模，使得语音合成更加自然和流畅。本实验就着训练成本方面的改进复现Parallel Tacotron 2并与Tacotron 2进行比较，对比其推理过程的算力需求和语音合成效果。

# 相关背景

Tacotron [3] 是一种先进的文本到语音生成模型，采用了Encoder-Attention-Decoder框架，并以RNN作为核心结构。与传统的统计参数TTS系统相比，这种端到端的模型减少了对特征工程的依赖，可以在模型的初始阶段就对某些条件进行控制，而不仅仅局限于个别组件中。这使得对音频属性的调节更加灵活，并且更容易适应新的数据。

但现在看来原始的Tacotron模型存在一些问题：其中一部分针对合成质量提出问题，主要包括合成质量问题，对数据数量和质量的需求问题等；另一部分则是针对Tacotron训练成本和复杂度进行优化。自从2017年谷歌推出了Tacotron模型之后，基于该模型的端到端框架，衍生出了多个变体，包括DurIAN [4]、Parallel Tacotron [5]、和, Parallel Tacotron 2 [1]等。

# 模型介绍

## 基本结构

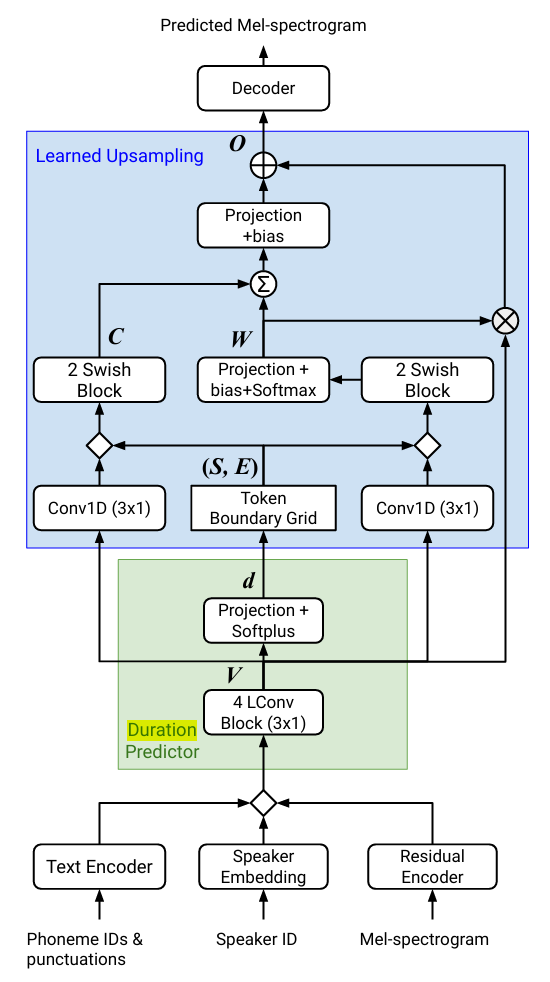
Parallel Tacotron 2 是一种非自回归神经文本到语音模型，它引入了完全可微分的时长模型，不需要监督时长信号。这个模型的基本框架如下：

**编码器：**三个编码器分别编码文本信息，说话人ID用于分辨多个说话人（很遗憾只能是有限个），和一个后验隐变量（训练）或者一个先验均值（推理）。

**持续时间预测器和可学习的上采样模块：**接收来三个编码器的输出，预测每个token的时长，即每个token应该对应多少个音频帧。并且会使用一个平均时长损失来确保预测的总时长与目标一致，并且所有操作都在实数域内进行以保证可微性。根据预测的token时长计算token边界，最后通过注意力机制将文本编码V上采样至O。这个过程明确了每个文本token对应的输出语音的时间，而且保证全程可微，便于优化。

**解码器：**接收上采样后的表示O作为输入。包含多个轻量级卷积模块来预测输出频谱图。

## 设计核心

**非自回归生成机制：**传统的自回归模型在生成每个时间步的输出时，依赖于前一个时间步的输出，换句话说，推理过程和训练过程都是串行进行的，这会导致生成以及训练的速度都比较慢。而 Parallel Tacotron 2 通过非自回归生成机制，能够并行生成所有时间步的输出，大大提高了生成速度和效率。在训练的过程中也同样如此，由于不存在类似RNN的串行机制，这极大的提高了训练的和推理的效率。

**可微分的持续时间建模：**Parallel Tacotron 2 在这里引入了可微分的持续时间建模(differentiable duration modeling)，可微分的持续时间建模是一种技术，用于在语音合成模型中更精确地预测和控制语音的持续时间。具体来说，模型可以在时间上连续建模，而不是依靠输出的token数量来控制持续时间，它允许模型在训练过程中直接优化持续时间参数，使得生成的语音更加自然和流畅。这种方法不仅提高了模型的整体性能，还使得生成的语音更加自然和流畅。

# 实验结果

## 实验环境

|  |  |  |
| --- | --- | --- |
| **Item** | **Tacotron 2** | **Parallel Tacotron 2** |
| Hardware | RTX 3080Ti Laptop | RTX 3080Ti Laptop |
| System | WSL: Ubuntu-22.04 | WSL: Ubuntu-22.04 |
| Driver | 556.13 | 556.13 |
| CUDA | 10.2 | 11.0 |
| conda | 24.9.2 | 24.9.2 |
| python | 3.7 | 3.7 |
| pytorch | 1.6.0 | 1.7.1 |
| torchaudio | None | 0.7.2 |

## 超参数

我仅使用他人提供的开源模型进行推理，并没有进行额外的微调或训练，因此此处的超参数仅有推理的超参数。基本上都是使用的官方或复现提供的参数[8,9,10]，未进行修改。

|  |  |  |
| --- | --- | --- |
| **Item** | **Tacotron 2** | **Parallel Tacotron 2** |
| batch\_size | 1 (default) | 1 (mode=“single”) |
| speaker\_id | 0 (default) | 0 (default) |
| max\_abs\_value | 4.0 (default) | None |
| power | 1.1 (default) | None |

## 实验结果

* + 1. **连读与弱读细节**

在对比Tacotron 2和Parallel Tacotron 2的过程中，尽管输入相同的一段文本，Parallel Tacotron 2生成的音频时长往往更短，大约少3%到10%左右。经过仔细聆听，我注意到Parallel Tacotron 2的音频显得更加连贯流畅，连读和弱读的现象更加明显，这说明了它在处理连读处的灵活性和准确性更强，更符合人类语音的打印规律。而这种表现差异，很可能得益于Parallel Tacotron 2采用的可微分持续时间建模方法，使得它对文本对应音频时长的把握更好，从生成的声音上说就是能做到该快的地方快、该慢的地方慢，从而更贴近人类语音的特点。

* + 1. **轻微失真问题改善**

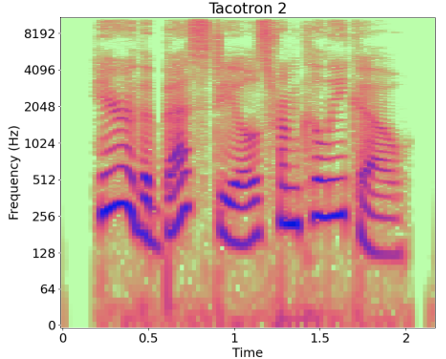
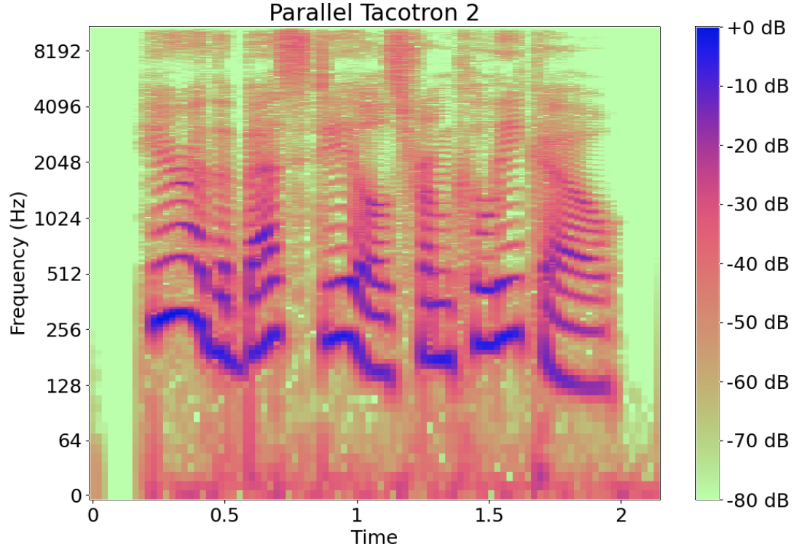
相比较于Tacotron生成的语音，Parallel Tacotron 2生成的语音中声音更加连贯。Tacotron在句子短暂停顿后的第一个发音，特别是句子开头处的发音，经常会出现一种轻微的不连贯类似气泡音的情况（或者描述为一种轻微的声音失真，类似音质不好造成的机械音的情况），而在Parallel Tacotron 2中我们能够听到更连贯更细腻的音频。直觉上的感受就是音质更好，听起来更清晰。

* + 1. **生成速度提高**

我对生成单个特定时长的音频所需要的时间进行了测试，这个结果仅是单次实验的结果，由于涉及到各种因素影响，以及没有重复实验，这种毫秒级别的时间测试不一定准确。但是这个实验结果可以对比看出，生成音频的长度与其算力需求是正相关的，而且Parallel Tacotron 2并行的方式的确起到了加速的作用。同时这两种方法也完全满足了实时语音生成的需求。

|  |  |  |
| --- | --- | --- |
| **Audio (second)** | **Tacotron 2 (second)** | **Parallel Tacotron 2 (second)** |
| 2 | 0.38 | 0.30 |
| 5 | 0.74 | 0.64 |
| 10 | 1.20 | 0.84 |

* + 1. **频谱图可视化**



## 不足与改进

**任意说话人支持：**虽然Parallel Tacotron 2 通过Speaker ID和Speaker Embedding支持了多说话人，但在实际应用中，如何能够zero-shot地实现发出一个新的说话人的声音仍然是一个大的挑战。到目前为止，GPT-SoVITS [7] 等更为先进的语音合成项目实现了这一功能。

# 结论

本实验调查并探究了Tacotron至今的语音合成的发展过程。并对比了Tacotron 2和Parallel Tacotron 2。详细说明了Parallel Tacotron 2的一些实现细节，并且复现了两个算法，使用开源的模型进行推理，在不同的文本上仔细对比二者的区别，并对结果进行了分析，同时也比较了生成速度，并且对生成的音频的频谱图进行了可视化。复现这两个算法使我对语音合成的理解进一步加深。

# 引用

[1] [Parallel Tacotron 2: https://arxiv.org/abs/2103.14574](https://arxiv.org/abs/2103.14574)

[2] [Tacotron系列模型介绍: https://zhuanlan.zhihu.com/p/706656935](https://zhuanlan.zhihu.com/p/706656935)

[3] [Tacotron: https://arxiv.org/abs/1703.10135](https://arxiv.org/abs/1703.10135)

[4] [DurIAN: https://arxiv.org/abs/1909.01700](https://arxiv.org/abs/1909.01700)

[5] [Parallel Tacotron: https://arxiv.org/abs/2010.11439](https://arxiv.org/abs/2010.11439)

[6] [Parallel Tacotron 2: https://google.github.io/tacotron/publications/parallel\_tacotron\_2/](https://google.github.io/tacotron/publications/parallel_tacotron_2/)

[7] [GPT-SoVITS: https://github.com/RVC-Boss/GPT-SoVITS](https://github.com/RVC-Boss/GPT-SoVITS)

[8]\_https://colab.research.google.com/github/r9y9/Colaboratory/blob/master/Tacotron2\_and\_WaveNet\_text\_to\_speech\_demo.ipynb

[9] <https://github.com/r9y9/wavenet_vocoder/blob/master/synthesis.py>

[10] <https://github.com/keonlee9420/Parallel-Tacotron2/blob/main/synthesize.py>