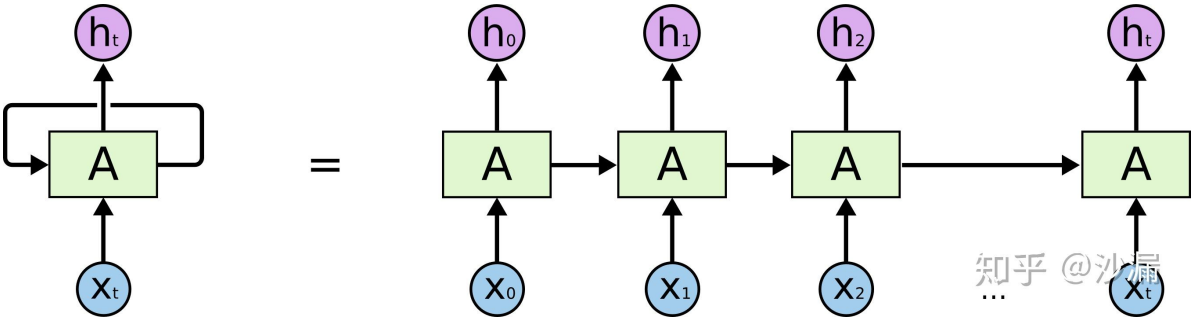
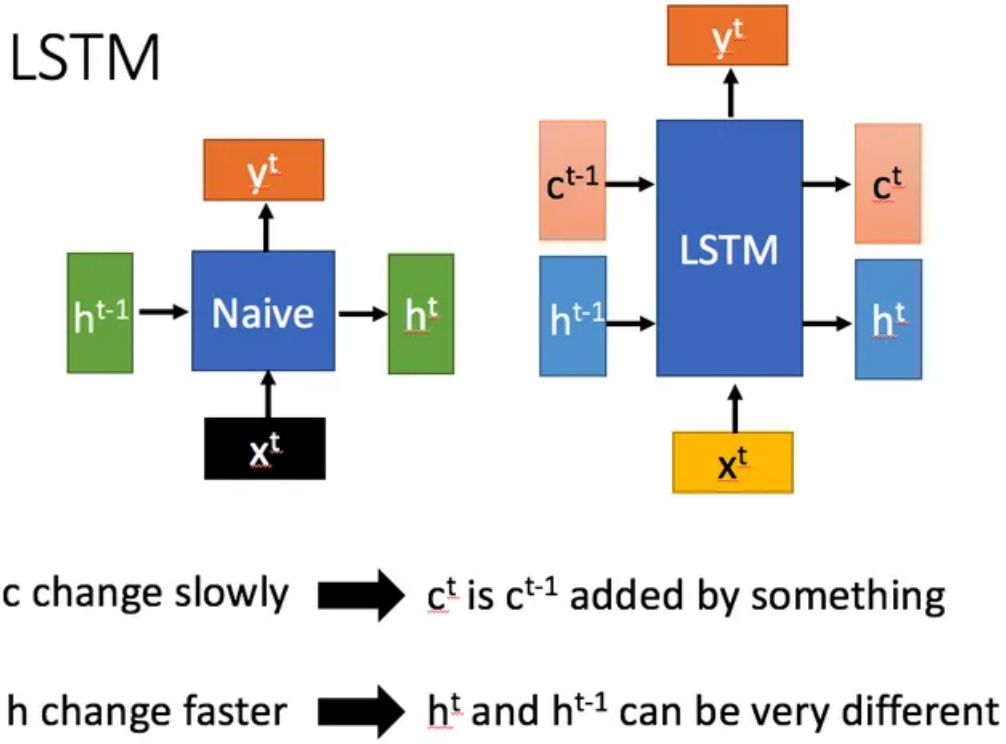
基础知识

RNN,LSTM,GRU

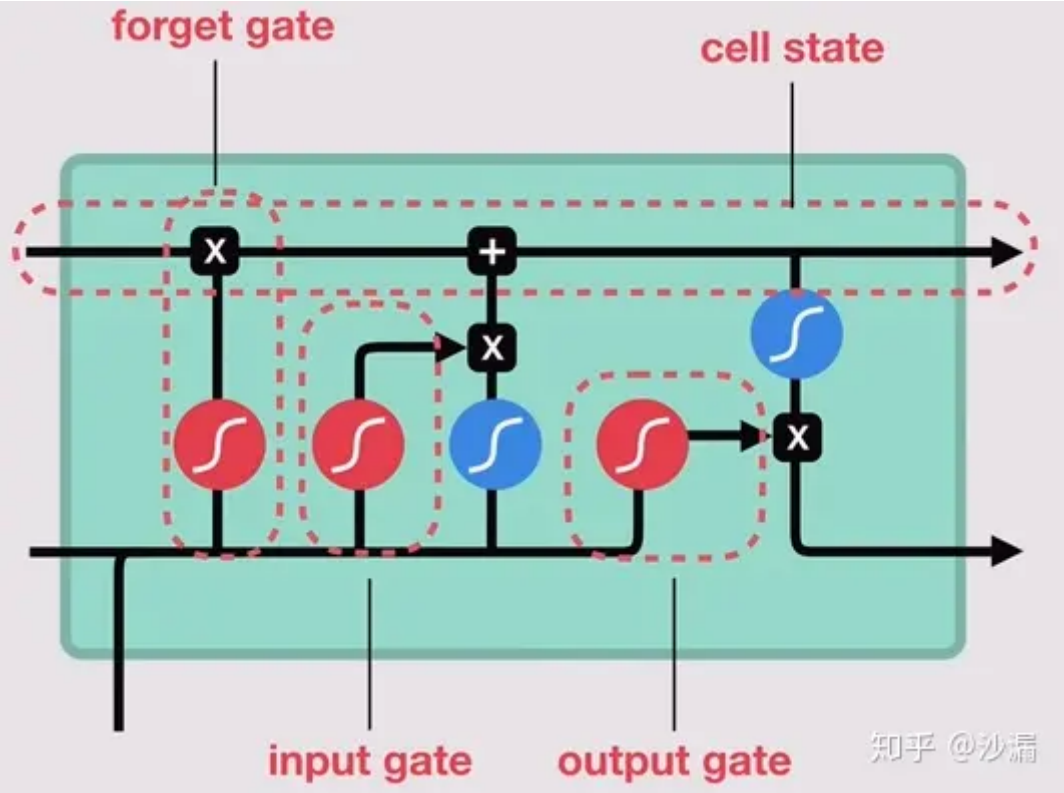
RNN的出现解决了上下文关联的问题，将上一时刻的输出送到下一时刻的输入中。每一时刻的输入包括输入信号以及上一时刻的隐藏信号，输出包括输出信号和隐藏信号。



RNN在理论上是保持着所有历史信息，但是越到后面，前面的文本信息就会越衰减，无法支持长期依赖。于是提出了LSTM，Long Short Time Memory，长短时记忆网络。



在LSTM中，采用专门的**“门”来引入或者去除上一时刻的细胞状态中的信息**，让信息选择性通过，是sigmod和按位的乘法操作，sigmod将输入信息映射到(0,1)来控制输入的“量”。



三个门：遗忘门，记忆门，输出门

- 遗忘门：决定细胞状态Ct-1要保留哪些信息。sigmod神经网络层（**和sigmod激活函数不是一个概念**）
- 记忆门：顾名思义用来“记忆”，决定新输入的信息xt和ht-1中哪些信息被保留并加到细胞状态Ct中。sigmod神经网络层+tanh神经网络层+按位乘法

  整合遗忘门的输出以及上一时刻细胞状态，相乘
- 输出门：sigmod神经网络层+tanh激活函数-10

sigmod神经网络层和sigmod激活函数不同

sigmod激活函数输入输出维度肯定是一样的因为只是一个数学函数。而sigmod层通常将输入数据进行线性变换然后通过sigmod激活函数进行非线性变换，输出维度是外部指定的。

```
class SigmoidLayer(nn.Module):
    def __init__(self,input_dim,output_dim):
        super(SigmodLayer,self).__init__()
        self.linear = nn.Linear(input_dim,output_dim)
        self.sigmod = nn.Sigmod()

    def forward(self,x):
        out = self.linear(x)
        out = self.sigmod(out)
        return out

#创建Sigmod神经网络层
sigmod_layer = SigmodLayer(10,1)  # 输入10，输出1
#将输入传递给Sigmod层
output = sigmod_layer([1,2,...,10])
```

为什么要是用激活函数

早期的感知机结构中，out=(input*weight)加和+bias，始终是线性变换。

激活函数是为了**引入非线性关系**，让神经网络可以表示复杂的模块和函数。常见的激活函数分为饱和激活函数和非饱和激活函数（包含函数就是在正负无穷处导数为0）

- **饱和激活函数**：sigmod[0,1]，tehn[-1,1]。sigmod导数小于0.25所以反向传播是梯度相乘，结果会趋向于0出现梯度消失状况。tanh同理
- **非饱和激活函数**：解决梯度消失问题；加快收敛速度。Relu=max(0,x)，当输入为正数不会出现神经元饱和，但是出现负数该神经元直接死亡。Leaky Relu,ELU等
- **sofimax[0,1]**，可以转为概率，常用于多分类任务，用于计算属于每个类别的概率。将一个K维的向量转换为K维的概率分布向量。不会导致梯度消失

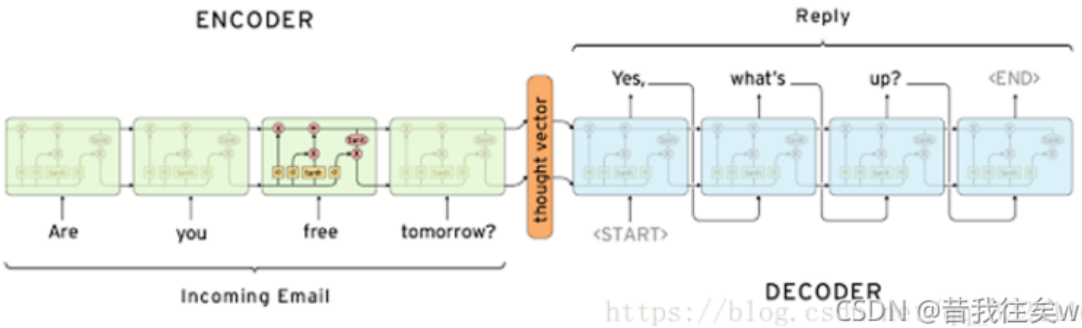
Seq2Seq模型

概念：序列到序列模型，就是输入一个序列，经过特定的方法**生成**另一个序列输出的方法。两个序列可以不等长。又叫**Encoder-Decoder模型**，即编码-解码模型。

产生原因

对于**RNN模型**来说，输入T个向量依次是x0,x1,...,xt，那么相应地会产生T个输出，**输入和输出长度是相同的**。如果输入和输出序列长度不同，那么是Seq2Seq模型。比如在机器翻译中，“hello”——>"你好"，在人机对话中，“你是谁’——>”我是chatgpt3“，长度都不同。

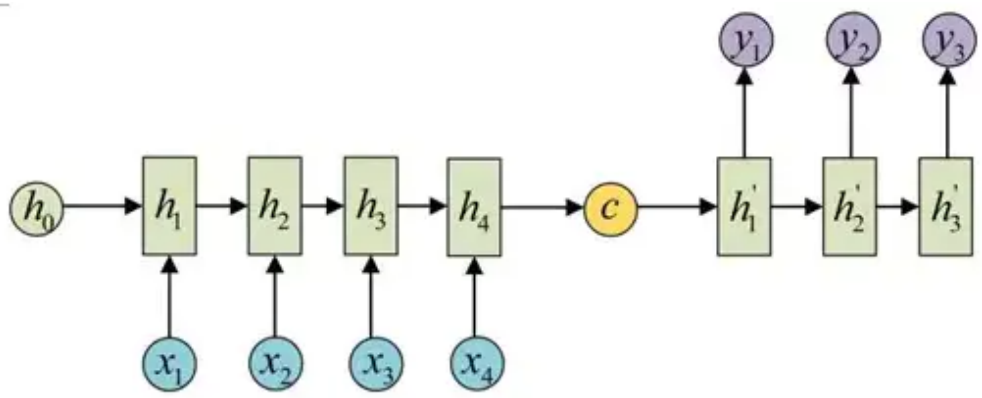
一个简单的对话问答示例：



结构

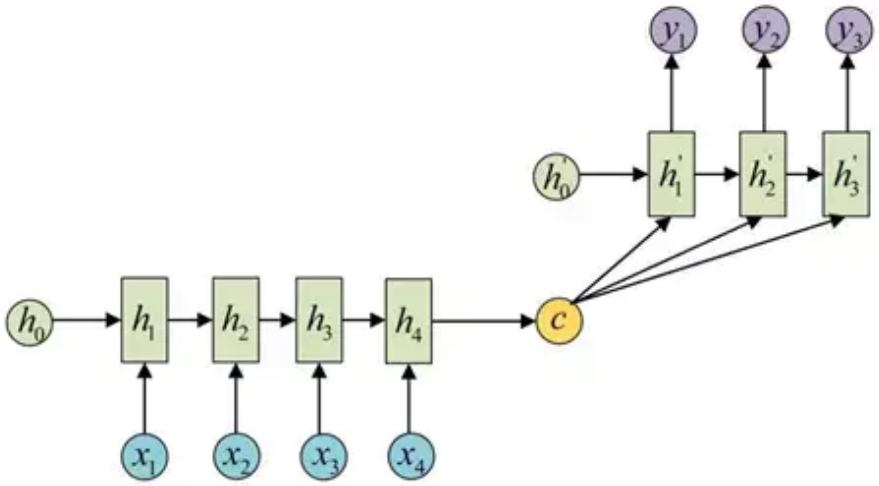
Seq2Seq由**编码器、解码器、和中间的编码向量**组成，编码器将任意长度地输入序列编码成**一个指定长度的编码向量**，解码器用于解码这个编码向量，生成输出序列。

- encoder-decoder都是**RNN框架**支持包含时间等前后关系的序列，常用的是**LSTM和GRU**。
- 解码器有两种编码方式：
  - 一是，编码向量是编码器的最后一个状态变换得到的，上一时刻的输出作为下一时刻的输入，进行预测下一个生成的词，直到生成 `<stop>`



语义向量只作初始化参数参与运算

- 二是，编码向量会参与编码器RNN的每个时刻运算。



语义向量参与解码的每一个过程　　CSDN @昔我往矣wood

- **训练原理**：实际上是在学习概率分布，根据前t个时刻的数据预测t+1时刻的数据，一般在输出层采用softmax函数得到各个词的概率，然后选取最大的作为输出结果。总结来说，目标是再先前给定t-1个单词的情况下预测序列中第t个单词是雌虎彪中第i个单词的概率
- 可以加入**注意力机制**

补充知识——从one-hoy到seq2seq

one-hot的问题：一个n维向量可以表示2^n个点，但one-hot实际上只使用了n个点，很稀疏。问题一是单词长度相同没有相对重要性，二是两两单词之间距离相同，没有任何差异性。

解决方法：映射到低维空间，一般是**左乘矩阵**，（M*N）(N*1) = (M*1)

Attention

对于编码器-解码器框架，当输入序列太长时，编码向量会信息过载，于是提出基于注意力机制的编码-解码模型。

编码器隐层输出的向量时密钥向量，解码器隐层生成的是查询向量

在解码的每一步解码时，都能有一个输入，对输入序列所有隐藏层信息h1，h2，……进行加权求和（显然对应的 Ht 和 ht 权重最大，其他的权重相对较小）。那这个"输入"就是query，

teacher forcing

有的seq2seq模型/RNN会使用teacher forcing机制进行训练。

为什么使用这个机制：早期RNN预测能力并不好，如何一个unit结果很差对后面的unit影响是很大的。比如要是`"[START]"`之后第一个词就预测错误，那整句话就跑偏了。

RNN实际上存在两种训练模式：

- **free-running mode**：常见的，上一个时刻的输入作为下一时刻的输出
- **teacher-forcing mode**：顾名思义，跟着老师学习，在训练过程中接收ground truth的输出y(t)作为t+1时刻的输入。缺点太依赖标签走不远，泛化能力不行，在测试集上表现不好

> 关于cross-domain和泛化能力generability
>
> **范围**：泛化能力关注的是模型在同一领域和任务上的未知数据上的表现。Cross-domain 能力关注的是模型在不同领域或任务上的适应性和性能。

> **技术手段**：泛化能力主要通过正则化、数据增强和交叉验证等方法来提高。Cross-domain 能力主要通过迁移学习、域适应、预训练和微调等方法来提高。
>
> **应用场景**：泛化能力在传统的监督学习任务中广泛应用，如图像分类、回归分析等。Cross-domain 能力在需要模型适应新任务或新领域的情况下更为重要，如多语言自然语言处理、跨领域图像识别等。

Teacher Forcing的缺点如何解决：

- 对于离散型输出（输出值是有限个可选值中的一个），如机器翻译等预测单词任务，使用**集束搜索beam search**，思想是：从输出状态开始，每一步会生成多个候选词，有一个参数叫**集束宽度beam width**，=3说明考虑三个预测得分最高的词
- 对于连续性输出（输出值是一个实数值），如房价预测股票预测任务等，**有计划地学习(Curriculum Learning)**，使用一个概率p去选择使用ground truth的输出y(t)还是前一个时间步骤模型生成的输出h(t)作为当前时间步骤的输入x(+1)。

  这个概率p会随着时间的推移而改变，这就是所谓的计划抽样(scheduled sampling)。训练过程会从force learning开始，慢慢地降低在训练阶段输入ground truth的频率。

关于backfard, backward, 优化器

- **前向传播**是指将输入数据通过神经网络层层传递，计算每一层的输出，直到最终生成模型的预测结果。
- **backward**是指根据预测结果与真实标签之间的误差，通过链式法则（链式求导）逐层计算梯度，并更新每一层的权重和偏置，以最小化误差函数。
- 优化器基于损失函数的梯度信息来调整模型的参数。

三者执行顺序：

前向传播：输入数据经过网络层，计算输出和损失。

反向传播：从损失出发，计算各层参数的梯度。

优化器更新：使用反向传播得到的梯度，更新模型参数。

关于forward()函数和inference()函数

forward()函数用于训练和评估阶段：

- 训练阶段

inference推断是用于**预测**阶段，相比forward会去掉dropout函数，完整充分利用神经元信息，在pytorch中通过设置`model.eval()`模式开启评估模式（关闭 dropout）

> 语音识别：SpeechBrain不适合初学者，初学者推荐wenet和espnet
>
> 语音合成：推荐amphian
>
> 中文数据集使用标贝，英文数据集使用LJSpeech
>
> fashspeech/tocach2上加语音人啥啥
>
> 用pytorch实现，tensorflow复用性不高

TTS

波形拼接就是比对输出的文本和语音库，匹配到语音波形后，将所有波形拼接起来。需要提前将录制好的语料放到数据库中，对数据库的要求很高，需要很大的数据库，但是拼接后的语音听起来都很自然。

SPSS不是直接拼接语音，而是先将文本转为声学参数，然后将声学参数通过算法转为波形。**文本分析**将输入的文本进行归一化（将特殊字符数字等转为文字，如1000转为一千或者一零零零）、字素音素转换、分词等，提取**语言特征**；**声学模型**先是使用匹配好的语言特征和**声学参数（如梅尔频谱或线性谱）**进行训练，然后将文本分析得到的语言特征转为声学参数；**声码器**将声学参数转为波形。更灵活，方便修改参数，数据库容量需求低

- 文本分析：提取发声、韵律等
- 声学模型：基于深度学习的生成网络可以直接将英文等文本转为频谱
- 声码器：纯信号处理，Griffin-Lim、STRAIGHT和WORLD；自回归深度网络模型，如WaveNet和WaveRNN；非自回归模型，如Parallel WaveNet、ClariNet和WaveGlow；基于生成对抗网络（Generative Adversarial Network，GAN）的模型，如MelGAN、Parallel WaveGAN和HiFiGAN。

前端处理与声码器都有通用的一些方案，针对不同任务的改进点主要在声学模型部分。比如引入深度学习模型DNN，学习从声音特征输出的映射函数，但是DNN忽略了声音的连续性，它假定每一帧都是独立采样的，后来就用RNN，使用所有可用的输入特征来预测每一帧的输出特征

Tacotron

面向端到端的语音合成

> front：
>
> CNN，RNN，LSTM，seq2seq
>
> Transformer中的自注意力机制，从通俗的生活实例中感性理解，从具体的任务如机器翻译并结合transformer源码落实到模型如何实现attention
>
> 感受野

加上wavernn应该会更流畅！

摘要

一个text-to-speech synthesis系统一般有多个阶段组成，例如一个文本分析前端text analysis frontend，一个声学模型acoustic model和一个声音合成模型audio synthesis module。构建这些组件通常需要大量的专业知识，并且可能包含错弱的设计选择。

> 脆弱的设计选择是什么意思？brittle design choices

在本文中，我们提出Tacotron，一个**end-to-end generative text-to-speech**模型，直接从字符合成语音。给定<text,audio>pair，模型可以通过**随机初始化从头开始进行训练**。我们提出一些关键的方法来让sequence-to-sequence框架在这个具有挑战性的任务上表现良好。Tacotron在US英语上实现了**3.82主观5-scale平均意见得分**，**在自然度naturalness上超过了生产参数化系统**。此外，因为Tacotron在**帧frame级别**上生产语音，所以它基本上比**样本级别sample-level自回归方法**更快。

> 通过随机初始化从头看开始训练？

> seuquence-to-sequence是什么框架？端到端框架就是直接从输入映射到输出不需拆分多个独立的步骤。seq2seq是一种典型的端到端框架，用于解决输入输出序列长度的不同的任务，如机器翻译、人机对话等，架构是编码器-解码器架构。

> 3.82 subjective 5-scale mean opinion score是啥意思？自然度？

> MOS是一种主观评分方法，满分是5，3.82处于第二档次(3.5-4.0)是"良"

> 生产参数化系统production parametric system是啥
>
> 传统的：就是那个基于HMM的语音合成，包括三四个步骤；现代参数化语音合成：tacotron系列，fastSpeech系列，wavenet

> frame-level和sample-level如何看

1. 介绍Introduction

现代TTS流水线比较复杂。例如，对于统计参数TTS来说有一个文本前端来提取多样的语言特征，一个中间模型，一个声学特征预测模型和一个复杂的signal-processing-based 声码器是很常见的。这些组件都是给予大量地领域专业知识，并且很难设计。他们也是单独训练的，所以来自每个组件的错误是会累计加重的。当前TTS设计的复杂度导致在构建一个新的系统时需要投入大量的工程工作。

因此，一个集合的端到端的TTS的优点就体现出来了，它可以用很少的人工标注的<text,audio>数据对进行训练。首先，这样一个系统**减少了艰辛的特征工程工作**，特征工程可能导致启发式错误错误的设计选择。其次，它允许在不同属性上加丰富的条件设置，比如**说话人或者语言，或者像观点sentiment之类的高水平特征。**这是因为条件设置可以发生在模型的一开始而不只是针对某个特定组件。类似地，针对新数据的修改会更容易。最后，一个简单的模型相比multi-stage模型**鲁棒性更强**，多阶段模型的每个组件的错误会不断累计。这些优势表明一个端到端的模型允许我们在大量**丰富、表现力强并且有噪音的现实世界数据中进行训练。**

TTS是一个大规模的颠倒问题：**一个高度压缩的源文本被"解压"成语音。**因为相同的文本可以关联不同的发音或者说话风格，这对于端到端模型也是一个特别困难的学习任务：它必须在signal level处理给定文本的多个变体。更重要的是，这和端到端语音识别、机器翻译不同，TTS输出是连续的，并且输出序列通常比输入更长。这些特征使得预测错误会快速累积。

> 这段话不是很理解

在本文中，我们提出Tacotron，一个端到端的基于带注意力范式的sequence-to-sequence的生成式TTS模型。我们的模型将字符作为输入，输出raw spectrogram，使用一些技巧来提高普通的seq2seq模型能力。给定<text,pair>数据对，Tacotron可以通过随机初始化从头开始训练。这并不需要音素级别的对齐phoneme-level aligment，所以它很容易调节以使用大量声学数据的抄本。借助一个简单的波形合成技巧，Tacotron可以产生3.82的平均意见得分（mean opinion score, MOS）在US 英语评估集合上，在自然度上超过了production paraetric system。

2. 相关工作

wavenet（2016年）是一个强大的语音生成模型。它在TTS任务上表现很好，但是因为它**sample-level autoregressive nature**（样本级别自回归本质）生成速度很慢。它还需要在已有TTS前端的语言特征上进行条件设置，所以并**不是端到端的**：他只是代替了声码器vocoder和声学模型。

另一个最近发展起来的模型是**Deep Voice（2017年）**，用一个相关的神经网络替换经典TTS流水线的每个组件。存在如下问题：

- 每个组件都是独立训练的，并且**改变系统去以端到端的方式进行训练并不容易。**所以很难说**seq2seq本身学到多少对齐能力。**
- 其次，为了训练好模型会使用一些技巧，这些技巧作者提到还会损伤韵律。
- 第三，它预测了声码器的参数作为中间特征表达，因此需要一个声码器
- 最后，该模型训练的是音素phoneme数据，所以实现结果多少会受到一些限制。

据我们所知，2016年Wang等人是最早**使用带有注意力机制的seq2seq框架**尝试端到端TTS的。然而，它需要一个预训练的隐藏HMM模型对齐器aligner来帮助seq2seq模型学习对齐alignment。

> 这两个模型不太懂是干啥，这里的对齐是指什么？

Char2Wav（**2017年**）是一个独立发展的端到端的模型，可以训练字符characters。然而，Char2Wav在使用一个SampleRNN神经声码器（2016年Mehri）之前也预测了声码器参数，但是Tacotron直接预测**raw spectrogram原始声谱图**。另外，seq2seq和SampleRNN模型修需要单独预训练，但是我们的模型可以直接从头开始训练。最后，我们对普通的seq2seq范式进行一些关键的修改。后面会展示，一个普通的seq2seq模型在字符级输入上表现并不好。

3. 模型架构

Tacotron的骨干是一个**带有注意力机制的seq2seq模型**。图一描述了模型，包括**一个编码器，一个基于attention的解码器，和一个后处理网络**。从高层面上说，我们的模型结构字符作为输入，并产生声谱图帧，这些帧之后会转换成波形。下面描述这些组件。

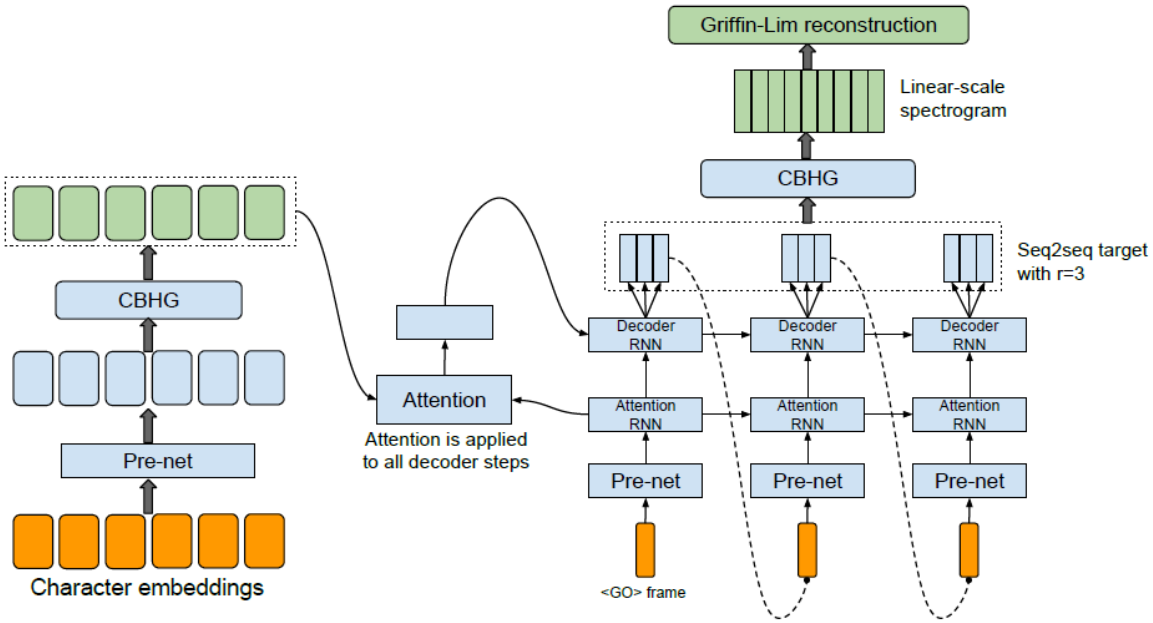


Figure 1: *Model architecture. The model takes characters as input and outputs the corresponding raw spectrogram, which is then fed to the Griffin-Lim reconstruction algorithm to synthesize speech.*

3.1 CBHG 模块

CBHG包含四个结构：**1-D卷积过滤器，高速网络highway networks，双向门控循环单元GRU，循环神经网络RNN。**

> 为什么是这四个结构

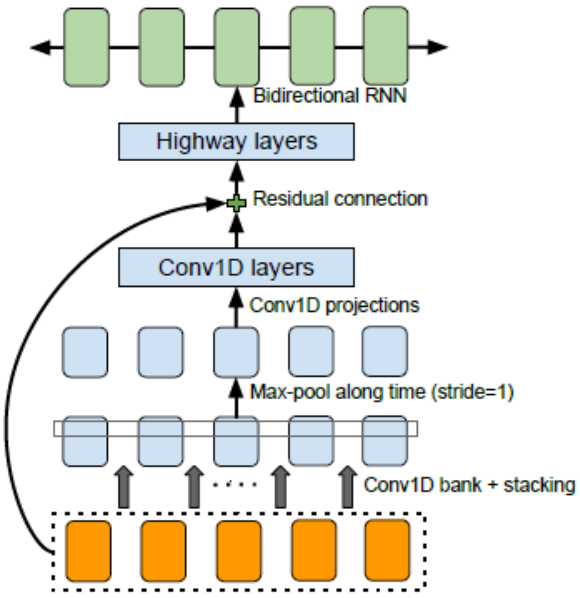


Figure 2: The CBHG (1-D convolution bank + highway network + bidirectional GRU) module adapted from Lee et al. (2016).

CBHG是一个强大的模块，用于从序列中提取特征表达。

输入序列首先通过**k组一维卷积核**进行卷积，第k组包括宽度为k的Ck个卷积核。这些过滤器显式地对局部和上下文相关的信息进行建模（类似于unigram,bigram和k-gram）。卷积操作的输出被堆叠在一起，之后再沿时间进行最大池化，以提高局部不变性。注意我们使用步长为1来维持初始状态的分辨率。我们进一步将处理好的序列传递给一些固定宽度的一维卷积核，他们的输出结果通过**残差连接**被添加到原始输出序列。批次归一化在所有卷积层都使用。

> 残差连接

卷积输出喂给一个**多层的高速网络**，来提取高纬特征。

最后，我们在顶层放置了一个**双向GRU RNN**，来从前向和后向上下文中提取序列特征。

CBHG是受到**机器翻译（2016年Lee）**启发，我们的工作和机器翻译（2016年Lee）主要的不同包括使用非因果关系的卷积核，批次归一化，残差连接和步长为1的最大池化。我们发现这些修改提高了泛化能力。

3.2 Encoder

编码器的目标是提取文本中的**鲁棒序列表达**`robust sequential representations`。编码器的输入是一个字符序列，其中每个字符都以独热码向量的形式表示并**嵌入embedded**到一个连续向量中。

> embedding嵌入，实际上是在做什么?
>
> 用向量表示对象的方法，如下图所示，embedding向量间的运算可以揭示性别关系

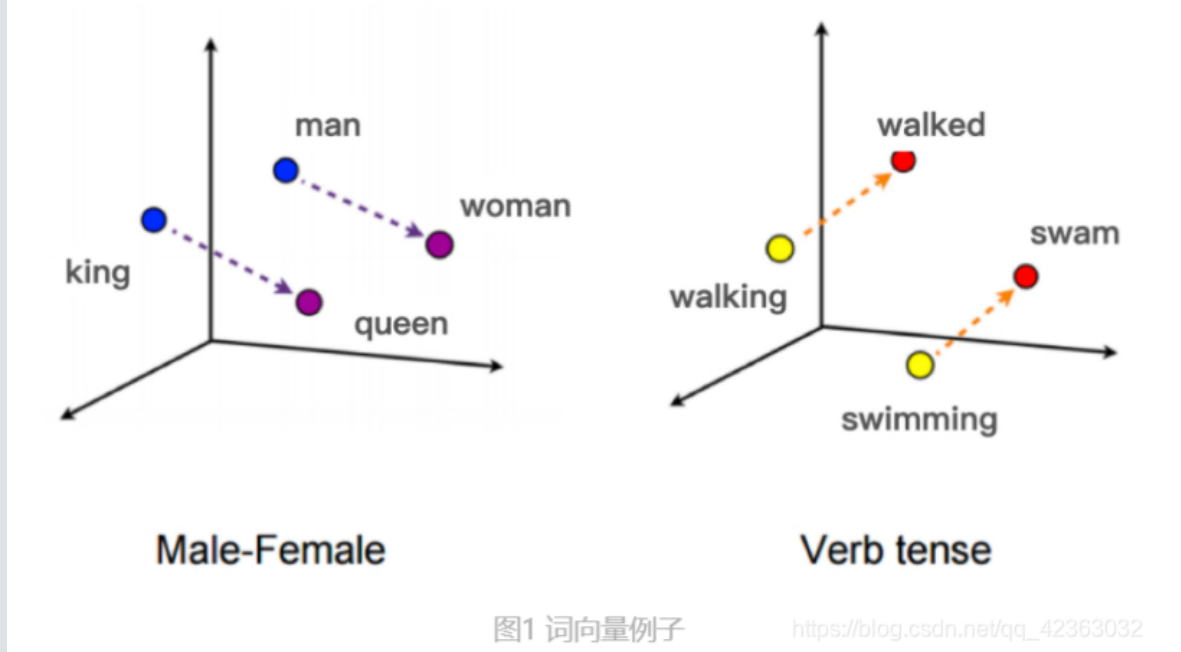


图1 词向量例子

我们然后对每个embedding（字符向量）施加一组非线性的转换，称为**"pre-net"**。我们使用一个带有drouput的瓶颈层bottleneck layer作为pre-net，帮助收敛和提高泛化能力。一个CBHG模块将pre-net的输出转换为最后编码器的注意力模块使用的特征表示。我们发现这个基于CBHG的编码器不仅能缓解过拟合，还比一个标准的多层RNN（参看链接网页上的合成语音样本）产生更少的发音错误。

3.3 Decoder

我们使用一个基于上下文的tanh注意力解码器（2015年Viny），其中一个**有状态的循环层会在每个时间步产生attention query**。我们连接上下文向量和注意力RNN单元的输出，形成解码器RNNs的输入。我们使用一个**带有纵向残差连接的GRUs堆栈来作为解码器**。我们发现残差连接加速收敛。

解码器的目标是很重要的设计选择。尽管我们可以直接预测原始频谱图，但这对于学习语音信号和文本之间的对齐而言，仍然存在高度的冗余（这是在本次任务中使用seq2seq的真实动机）。因为这个冗余，我们为seq2seq解码和波形合成使用了不同的目标。

> 冗余？是因为频谱图包含的声音信息太多了吗？

语音合成作为一个可训练或者可确定的逆向过程，只要能提供足够的语音可理解性和韵律信息，seq2seq的目标输出就可以被大幅度压缩。我们使用**80-bandmel-scale spectrogram（带宽为80的梅尔刻度声谱图）**作为目标输出，尽管更少的带宽band或者更简洁的目标输出例如cepstrum也能使用。我们使用一个**后处理网络**（下面会讨论）来从seq2seq目标输出转为波形。

我们使用一个简单的**全连接输出层**来预测解码器的目标输出。我们发现的一个重要技巧是在每个解码步骤可以**同时预测多个非重叠的输出帧。每预测r个帧就将总的解码步骤缩小n倍**，这减少了模型大小、训练时间和推理时间。更重要的是，我们发现这个技巧可以大幅度增加收敛速度，这可以通过**注意力机制学习对齐关系**的速度更快且更稳定来衡量。这可能是因为相邻的音帧具有相关性，并且每个字符通常对应多个音帧。依次输出一个帧迫使模型在多个时间步上关注同一个输入token；输出多个音帧运行注意力在训练中更早地向前移动。一个类似地技巧2016年Zen等人也使用了，但是主要是加速推断。

第一个解码步骤是在一个**"全零帧"**上进行调节，代表`<GO> frame`。在推断时，在第t步解码步骤中，将r个预测结果中最后一帧作为解码器第t+1步的输入。注意，选择最后一帧作为输入只是一种选择——也可以使用一组r个帧的全部作为下一步的输入。在训练中，我们总是选取每个第r帧作为解码器的输入。输入帧传递给pre-net，正如在编码器中的处理一样。因为我们没有使用像**预排程采样**（2015）之类的技巧（我们发现这会损害音频质量），所以pre-net中的dropout对模型泛化时很关键的，因为它为解决**输入分布中的多形态问题**提供了**噪音源**。

3.4 Post-Processing Net and Waveform Synthesis

正如上面提到的，后处理网络的任务时将seq2seq的目标输出转换为可以合成波形的目标表达。因为我们使用**Griffin-Lim**作为合成器，后处理网络要学习的是**如何预测在线性频率刻度上采样的频谱幅度**。构建后处理网络的另一个动机时它可以看到全体解码序列。和seq2seq相比，seq2seq总是从左至右运行，而后处理网络可以**获取前向和后向信息用来纠正单帧的预测错误**。在这次工作中，我们使用一个CBHG模块作为后处理网络，**尽管一个更简单的架构看你表现会更好。**

> 那为什么还要使用CBHG?

后处理网络的概念是高度通用的。它可以用于预测不同的目标输出例如声码器参数，或者就像WaveNet-like神经声码器（2016年van）一样用于直接合成波形样本。

我们使用Griffin-Lim算法来从预测的声谱图合成波形。我们发现在输入GL之前，将**预测频谱的振幅提高1.2倍**，可以减少人工痕迹，这可能是因为它的**谐波增强效果**。我们观察到，GL在**50次迭代后会收敛**（事实上，大约30次迭代差不多就足够了），这个速度相当快。我们在tensorFlow中实现GL算法，因此它也成为模型的一部分。尽管GL是可导的（**它没有可训练的参数**），但我们没有在这个工作上**设计什么loss**。我们强调，我我们选择GL是为了简单；尽管它已经产生了很好的结果，但是我们也在研究发展一个快速并且高质量的可训练的声谱-波形转换器。

4. 模型细节

5. 实验

6. 讨论Discussions

我们提出Tacotron，一个集成的端到端的生成式TTS模型，输入一个字符序列输出相应的spectrogram。

借助一个非常简单的波形合成模型，它在US English数据上实现了3.82 MOS得分，在自然度上超过了production parametric system。

Tacotron是基于帧frame-based，所以接口基本上比sample-level autoregressive methods更快。

和之前的工作不同，Tacotron不需要人工标注语言特征也不需要像HHM aligner这样复杂的组成。

> HMM aligner是啥

它可以通过随机初始化从头进行训练。我们进行了简单的文本归一化，不过最近在文本归一化学习上的进步（2016年SJ）表明这在未来可能并不是必须的。

我们还需要调研模型的多个方面；很多早期的设计决策一直保持原样。我们的输出层，注意力模型，损失函数，以及Griffin-Lim-based波形合成器都有待提高。例如，总所周知Griffin-Lim 输出可能有人工痕迹。我们现在聚焦于快速且高质量的基于神经网络的spectrogram inversion.

Tacotron2

摘要

本文描述Tacotron2，一个神经网络架构，用于从文本直接合成语音。系统由一个**循环seq2seq特征预测网络**组成，将**字符嵌入映射到梅尔频谱**，后面是一个**修改过的WaveNet模型作为声码器**从这些频谱图合成时域波形。我们的模型实现4.53的MOS得分，与专业录制的语音4.58旗鼓相当。为了验证我们的设计选择，我们对我们的系统关键组件进行消融实验并评估了**使用梅尔频谱作为WaveNet的条件输入**，而

不是语言学特征、时间特征和F0特征的影响。我们进一步展示了使用这个简单的声学中间特征表示可以大大减小WaveNet架构的大小。

1. 介绍

【历史发展】从文本生成神经语音一直是这几十年研究中的挑战。这一领域的主导技术在随着时代的发展不断迭代更新。拼接式合成，通过单元选择将预录制的波形小单元拼接起来的方法，多年来都是领域里的前沿技术。参数语音合成是直接生成语音特征的平滑参数，之后通过声码器合成，解决了拼接合成技术中存在的**边界伪影(boundary artifacts)**问题。然而，这些系统产生的语音和人类语音相比经常听起来很沉闷并且不自然。

WaveNet，一个生成时域波形的模型，产生的音频质量开始和真实的人来语音媲美并且已经用在一些完整的TTS系统中。WaveNet的输入（语言特征，预测的对数基频F0，和音素时长），需要显著的领域专业知识生成，涉及复杂的文本分析系统和鲁棒分词器（发音指南）。

Tacontron，一个seq2seq架构用于从字符序列生成magnitude spectrograms，简化了传统语音合成流水线通过用一个简单的单独用数据训练过的神经网络代替在这些语言和声学特征生成过程。为了用语言编码器来分析或合成magnitude spectrograms，Tacotron使用Griffin-Lim树算法来进行相位估算phase spectrograms，后面紧跟着是一个短时傅里叶变换。正如作者指出，这只是为未来神经声码器方法预留的一个占位符，因为GL会产生特有的伪影并且音频质量比WaveNet等方法还要低。

在本文中，我们描述了一个统一的、完全的神经方法进行语言合成，结合了过去方法中的精华：**一个seq2seq Tacotron-style模型用于生成梅尔频谱，接着是一个修改过的WaveNet声码器。**直接在正则化字符序列号和相应的语音波形上进行训练，我们的模型学习合成自然的声频，很难和真实的人类语音进行区分。

Deep Voice3描述一个相似的方法。然而，不像我们的系统，它的自然度无法和人类语音相媲美。**Char2Wav**描述了另一个相似的方法，在端到端TTS中使用一个神经声码器。然而，他们使用不同的中间特征表达（传统的声码器特征），并且他们的模型架构和我们的显然不同。

2. 模型架构

如图一所示，我们的模型由两部分组成：（1）一个带有注意力机制的循环seq2seq特征预测网络，用于从输入的字符序列预测梅尔频谱帧序列（2）一个WaveNet修改后的变体，在预测的梅尔频谱帧上生成时域波形样本。

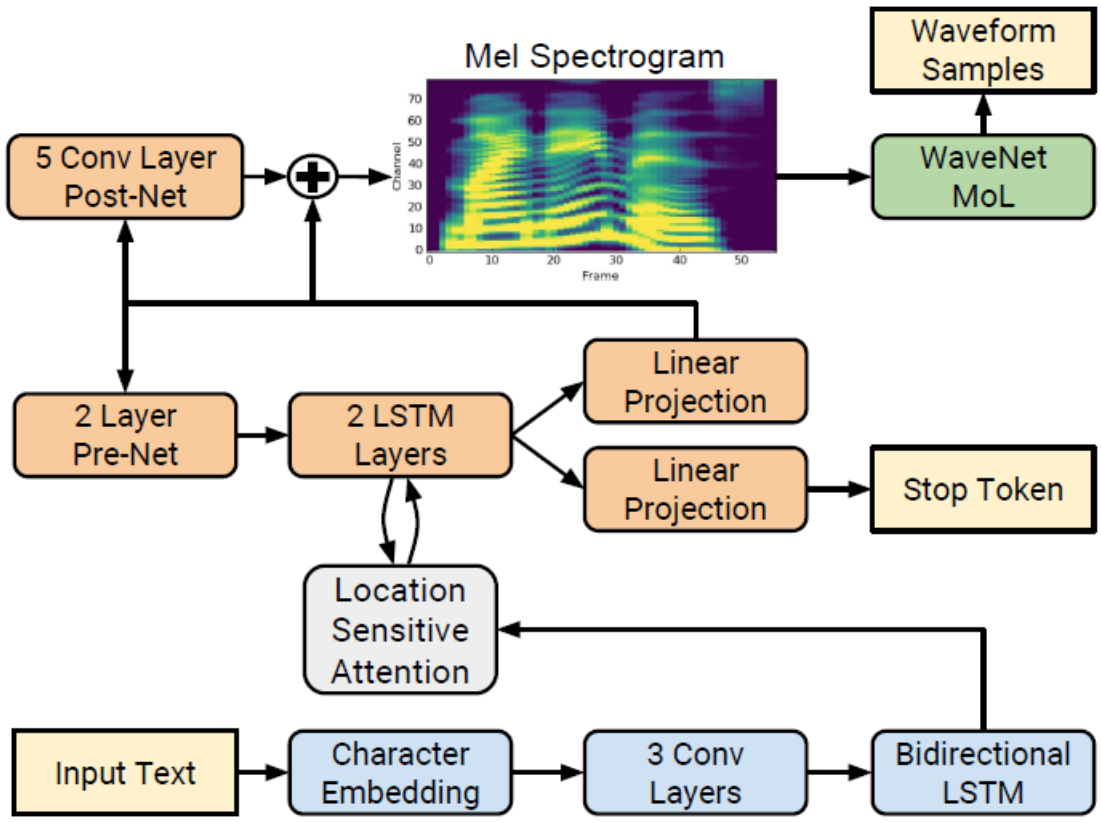


Fig. 1. Block diagram of the Tacotron 2 system architecture.

2.1 中间特征表达

在这次工作中，我们选择一个低维度的声学特征表达：mel-frequency spectrograms，来建立两个组件之间的桥梁。**梅尔频率声谱图通过对时域波形进行计算很容易得到，使用这样一个表征，为我们独立训练两部分组件提供了可能。**梅尔频谱比波形样本更平滑，并且由于其每一帧都是对相位不变的，所以更容易**用均方误差损失MSE进行训练。**

梅尔频率声谱图和线性频率声谱图（即短时傅里叶变换的振幅）是相关的。从对人类听觉系统的响应测试中得到启发，**梅尔频谱是对短时傅里叶变换的频率轴施加一个非线性变换，用较少的维度对频率范围进行压缩变换得到的。**使用这样的听觉频率刻度可以突出低频降低高频，突出低频有助于语音的可理解性，而高频通常由摩擦和其它噪音爆破而成，通常不需要高保真度的建模。因为这些属性，从梅尔刻度衍生的特征几十年间都被用作语音识别的唯一特征表达。

线性声谱图抛弃了相位信息（因此是有损的），而像Griffin-Lim [14] 这样的算法可以对抛弃的相位信息进行估计，用一个短时傅里叶逆变换就可以把线性声谱图转换成时域波形。梅尔声谱图抛弃的信息更多，这对于**逆向波形转换**是一个挑战。但是，对比WaveNet中使用的语言学和声学特征，梅尔声谱图更简单，是音频信号的更低层次的声学表征，因此使用类似WaveNet的模型构造神经声码器时，在梅尔声谱图上训练语音合成应该更直截了当。我们将会展示用WaveNet架构的修订版从梅尔声谱图可以生成高质量的音频。

2.2 频谱预测网络

与Tacotron一样，mel频谱图通过短时傅里叶变换（STFT）计算，使用50毫秒的帧大小，12.5毫秒的帧跳跃和Hann窗口函数。我们尝试了5毫秒的帧跳跃，以匹配原始WaveNet中条件输入的频率，但相应的时间分辨率的增加导致显著的发音问题。

我们使用覆盖125 Hz到7.6 kHz的80通道mel滤波器组将STFT幅度转换为mel尺度，然后进行对数动态范围压缩。在对数压缩之前，滤波器组的输出幅度被剪裁到0.01的最小值，以限制对数域中的动态范围。

声谱图预测网络中，包含一个编码器和一个引入注意力（attention）机制的解码器。编码器把字符序列转换成一个隐层表征，继而解码器接受这个隐层表征用以预测声谱图。输入字符被编码成512维的字符向量，然后穿过一个3层卷积，每层卷积包含512个5 x 1的卷积核，即每个卷积核横跨5个字符，后接批标准化（batch normalization）[18]和ReLU激活函数。像Tacotron中一样，卷积层会对输入字符序列的大跨度上下文（例如N-grams）进行建模。最后一个卷积层的输出被传递给一个双向[19] LSTM [20]层用以生成编码特征，这个LSTM包含512个单元（每个方向256个单元）。

构建一个**注意力网络**(attention network)用以消费编码器的输出结果，编码器的每次输出，注意力网络都将编码序列归纳为一个定长上下文向量。我们使用[21]中的**位置敏感注意力机制**，

2.3 WaveNet声码器

3. 实验&结果

3.1 训练设置

我们的训练过程包括，首先**单独训练特征预测网络，然后基于特征预测网络的输出，来训练修改版的WaveNet。**

- 训练特征预测网络：最大似然训练规程（在解码器端不是传入预测结果而是传入正确的结果，这种方法也被称为teacher-forcing），在单个GPU上使用batch size=64.
- 训练修改后的WaveNet：基于特侦预测网络中的`ground truth-aligned`（真实标签）预测结果来训练，就是说网络是以teaching force模式运行，**所预测的每一帧是基于编码的输入序列以及所对应的前一帧标定数据频谱。这保证了每个预测帧与目标波形样本完全对齐。**

> teacher forcing? 训练RNN或seq2seq模型的技术，**在训练过程中，模型在每一步生成下一步的输入时，不适用模型自己的输出，而是使用真实的目标序列。**这里的输入输出就是"帧"
>
> 与之相对的是，**自回归**训练机制。自回归体现在测试过程，但是如果用于训练会很难收敛。

> "ground truth"（简称为 GT）指的是在模型训练和评估过程中使用的真实标签或参考数据。

> Tacotron2的mel频谱预测网络是一个自回归网络，为什么论文中还说训练是teacher forcing呢，这二者不冲突吗?
>
> 自回归模型在生成语音序列时，依赖于之前生成的输出。每一步的输出都是基于之前所有已生成的内容来预测的。非自回归模型是在生成序列时，每个时间步的输出是独立生成的，不依赖于之前生成的输出。
>
> teacher forcing只是一种训练方法，主要用于克服自回归模型在训练早期阶段可能出现的错误累积问题。
>
> 它们是相辅相成的：自回归模型定义了生成序列的方式，而教师强制则是一种训练策略，帮助模型更好、更快地学会生成高质量的序列。

3.2 评估

MOS得分。

- 表一和其它TTS模型进行比较，结果最优
- 比较tacontron2合成语音和标记真实语音， -3（合成结果比标定真实语音差很多）到3（合成结果比标定真实语音好很多），最后得分−0:270 ± 0:155，说明评分者在很小的程度上更喜欢标定真实语音。偶尔的发音错误是更喜欢真实语音的主要原因。

3.3 消融实验

3.3.1 预测特征 vs 标定真实数据

因为两个组件是分开训练的，WaveNet组件要依赖于前一个组件的特征预测结果，所以可以使用从标定真实数据抽出的梅尔声谱图来训练WaveNet。表2展示了可能性。

3.3.2 线性声谱图

了与梅尔声谱图的效果做对比，我们训练特征预测网络使其预测线性频率声谱图，这样就可以用Griffin-Lim算法对声谱图进行逆变换。

- WaveNet比Griffin-Lim算法生成的语音质量高很多
- 使用线性声谱图或者梅尔声谱图却没有太大区别

3.3.3 后处理网络

由于预测特征帧在被解码前不能被使用，我们在解码后使用了一个卷积后处理网络把过去帧和未来帧都包含进来以改善未来帧的预测。然而，因为WaveNet已经包含了卷积层，可能会质疑使用WaveNet声码器的话，后处理网络是否还有必要。

比较了使用和不使用后处理网络的结果，不使用后处理的MOS评分只得到了4:429 ± 0:071，而使用后处理的MOS评分是4:526 ± 0:066

3.3.4 简化WaveNet

a large receptive field size is not an essential factor for audio quality. 对于语音质量来说一个大的感受野并不是必须的

4. 结论

本文描述Tacotron2，一个完全神经TTS系统，结合一个带有注意力机制的seq2seq循环网络用于预测梅尔频谱和一个修改过的WaveNet声码器。结果系统合成带有Tacotron-level的韵律和WaveNet-level的音频质量。这个系统可以直接从数据中进行训练，不需要依赖于负责的特征工程，并且达到最前沿技术的声音质量，和自然人类语音十分接近。

模型复现

Apex：NVIDIA 开发的一个 PyTorch 扩展库，用于提高深度学习训练的性能和效率。它提供了一些高级功能，特别是在**混合精度训练和分布式训练**方面

```
Train loss 0 47.335262 Grad Norm 13.342476 3.15s/it
Validation loss 0: 32.986515
Traceback (most recent call last):
  File "train.py", line 290, in <module>
    train(args.output_directory, args.log_directory, args.checkpoint_path,
  File "train.py", line 246, in train
    validate(model, criterion, valset, iteration,
  File "train.py", line 146, in validate
    logger.log_validation(val_loss, model, y, y_pred, iteration)
  File "/root/autodl-tmp/tacotron2-master/logger.py", line 27, in log_validation
    self.add_histogram(tag, value.data.cpu().numpy(), iteration)
```

```
  File "/root/miniconda3/lib/python3.8/site-
packages/torch/utils/tensorboard/writer.py", line 441, in add_histogram
    histogram(tag, values, bins, max_bins=max_bins), global_step, walltime)
  File "/root/miniconda3/lib/python3.8/site-
packages/torch/utils/tensorboard/summary.py", line 320, in histogram
    hist = make_histogram(values.astype(float), bins, max_bins)
  File "/root/miniconda3/lib/python3.8/site-
packages/torch/utils/tensorboard/summary.py", line 344, in make_histogram
    cum_counts = np.cumsum(np.greater(counts, 0, dtype=np.int32))
TypeError: No loop matching the specified signature and casting was found for
ufunc greater
```

```
cum_counts = np.cumsum(np.greater(counts, 0, dtype=np.int32))
改为:
cum_counts = np.cumsum(np.greater(counts.astype(np.int32), 0))
```

说话人转换

```
p237 16epochs 32batchsize
Epoch: 15
Train loss 120 0.351106 Grad Norm 1.959465 2.88s/it
Train loss 121 0.308219 Grad Norm 0.968583 3.55s/it
Train loss 122 0.364125 Grad Norm 1.135402 2.68s/it
Train loss 123 0.223638 Grad Norm 0.608693 3.60s/it
Train loss 124 0.278719 Grad Norm 1.328667 2.82s/it
Train loss 125 0.447735 Grad Norm 1.452149 1.68s/it
Train loss 126 0.434743 Grad Norm 1.381348 1.87s/it
Train loss 127 0.426162 Grad Norm 3.291003 1.70s/it
```

Train loss 127 0.426162 Grad Norm 3.291003 1.70s/it

第二版:

```
Train loss 3000 0.186070 Grad Norm 0.507803 2.30s/it
Validation loss 3000:  0.556794
```