本周又看了一遍Wavenet源代码

WaveNet通过直接为音频信号的原始波形建模，一次为一种音频样本建模。同生成听起来更为自然的语音相同，使用原始波形意味着WaveNet能够为任意类型的音频建模，包括音乐。

在训练时间段内，输入序列是从人类说话者记录得来的真实波形。在训练之后，对网络取样，以生成合成话语。在取样的每一个步骤中，将从由网络计算得出的概率分布中抽取数值。所抽取的数值随后被反馈到输入信息中，这样便完成下一步新预测。像这样每做一次预测就取样会增加计算成本，但是我们已经发现，这样的取样方法对于生成复杂且听起来真实的音频是至关重要的改善最优文本-语音转换模型

过程：

1.WaveNet神经网络架构直接生成原始音频波形，在文本到语音和一般音频生成中显示出出色的效果

2.网络模拟条件概率以生成音频波形中的下一个样本，给定所有先前的样本以及可能的附加参数。

3.在音频预处理步骤之后，将输入波形量化为固定的整数范围。然后，整数幅度被一次热编码以产生张量的形状(num\_samples, num\_channels)。

4.只能访问当前和先前输入的卷积层然后减小通道尺寸。

5.网络的核心构造为一堆因果扩张层，每层都是扩展卷积（卷积孔），只能访问当前和过去的音频样本。

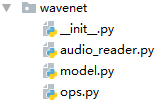
6.所有层的输出被组合并通过一系列密集的后处理层延伸回原始数量的通道，随后是softmax函数以将输出转换成分类分布。

7.损失函数是每个时间步长的输出与下一个时间步长的输入之间的交叉熵。

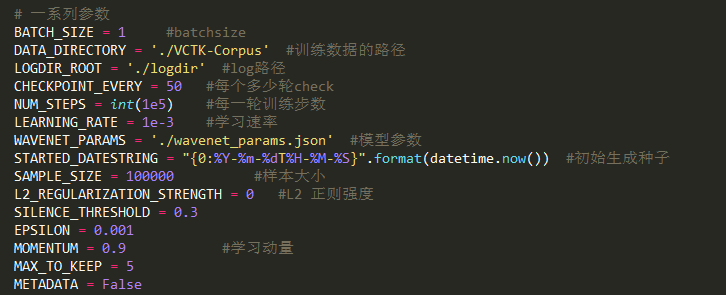
8.在该存储库中，可以在model.py中找到网络实现。

关键的文件为train.py，generate.py和wavenet文件夹。train.py为训练代码，generate.py为生成代码。wavenet文件夹包括了所需的模型，语音读取，以及其它功能类和方法。

wavenet文件夹包含文件如图所示：



Train.py解析



模型保存/模型加载

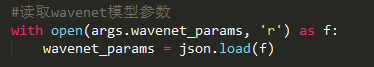




Main方法

包含：

1.读取wavenet模型参数



2.建立tensorflow的coodrdinator



3.从VCTK数据集生成input



4.建立wavenet模型



5.训练并保存模型

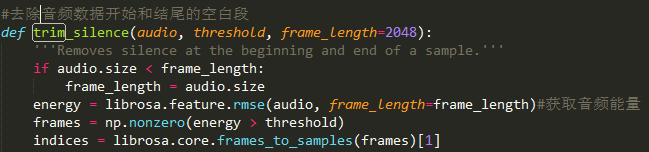


Main 方法使用了Audio\_reader.py和model.py中的类

audio\_reader.py包含了四个方法

（find\_files()，load\_generic\_audio()，load\_vctk\_audio()，trim\_silence()）和一个类 AudioReader()。

四个方法中，需要关注一下的是trim\_silence()方法。该方法是去除音频数据开始和结尾的空白段。

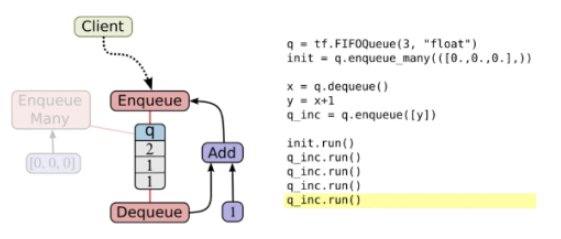


AudioReader()类

功能：把预处理好的音频数据打包成tensorflow queue

队列的使用包括几个步骤：

一，建立队列；二，初始化队列；三，队列的入栈和出栈



model.py解析

该部分是源码最精华的部分，包括了建立网络模型和语音生成器相关函数。因为内容较多，同时生成器相关函数和建立网络模型相关函数大同小异,此处只详解网络模型建立相关函数。

model.py包含两个函数一个类。其中，create\_variable()和create\_bias\_variable()功能分别为创建/初始化权值和bias的函数，很简单。

WaveNetModel类中关键函数有：

创建变量函数\_create\_variables()

创建因果卷积函数\_create\_causal\_layer()

创建扩大卷积函数\_create\_dilation\_layer()

建立网络模型函数\_create\_network()以及loss()函数

创建变量函数\_create\_variables()

该函数创建了模型建立所需的所有变量（因果/扩大卷积层以及后处理层所需的变量），并存为字典待使用。

因果卷积函数\_create\_causal\_layer()

该函数功能是建立因果卷积。函数调用了ops.py 文件中的causal\_conv()函数。该因果卷积的实现方法是采用将输出偏移几步来实现，具体采用的是tf.pad()方法来实现偏移。

创建扩大卷积\_create\_dilation\_layer()

该函数实现扩大卷积层，同时在该层创建了residual 和skip connection，让模型更快收敛

建立网络模型函数\_create\_network()

该函数采用前面的\_create\_dilation\_layer()建立网络模型。在因果卷积后面，加上了后续处理层（postprocessing layer）。后续处理层结构为：Perform (+) -> ReLU -> 1x1 conv -> ReLU -> 1x1 conv。

loss()函数

该函数首先将输入语音数据进行\mu -law编码(mu\_law\_encode())后再使用one-hot编码。loss函数采用的是tf.nn.softmax\_cross\_entropy\_with\_logits()。

generate.py解析

这部分代码用于模型语音生成。

WaveNet项目，这是一种全面卷积（Convolutional），基于概率，可自动回归的深度神经网络。根据DeepMind的介绍，该项目可以用比现有最好的文字转语言（TTS）系统更棒的效果通过音频和声音合成更自然的语音和音乐。

DeepMind认为，原有模式极大依赖通过一个输入源，或一个录音者生成的大容量音频数据库，WaveNet依然保留了这种模式，并将其作为一组参数，可根据新的输入结果对原有模式进行修改。这种方法也叫做参数化（Parametric）的TTS实现，可通过支持参数的模型生成在音调或语调等特征方面有所差异的语音，随后这些语音还可通过模型进行进一步的完善。相比以往的方法使用预先生成的原始音频片段对模型进行训练，WaveNet的Phoneme可调整字词和句子的顺序参数，生成更有意义的词语和句子结构，并可独立于有关声调、声音质量，以及音素语调的参数进行调整。借此WaveNet可以生成连续的语言类声音，并通过语言结构为这些声音赋予相关的含义。

疑问：在看model.py时，wavenet的输入是音频波形，而TTS输入的是文本信息，并没有很懂wavenet是如何应用到TTS的系统中的。