

机器学习实验报告

题目：基于Wavenet的音乐合成

姓名：黄琛权\_

学号：17251007

指导教师： 陈一帅

学院： 电子信息工程学院\_

**2020.5.22**

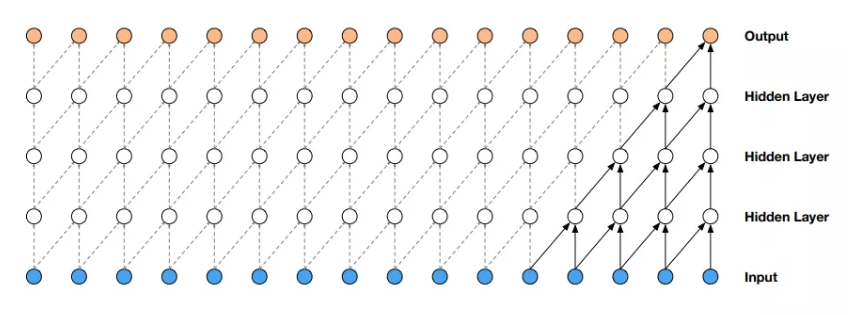
**一、实验环境**

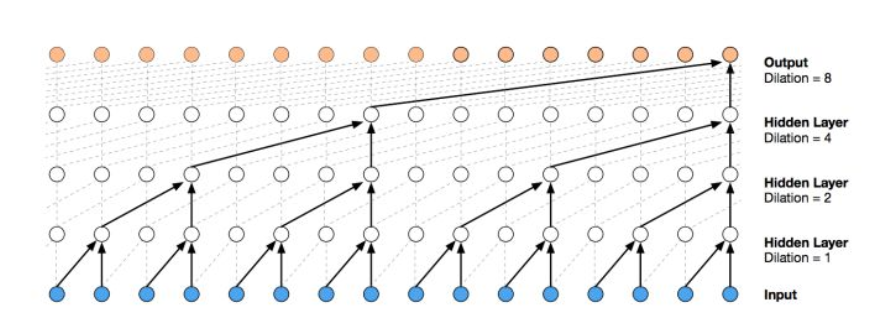
这里的实验环境有两种选择：

第一种选择：在计算机上提前下载Python 3.7（安装Magenta）、Tensorflow、Nvidia Collective multi-GPU Communication Library（NCCL）。

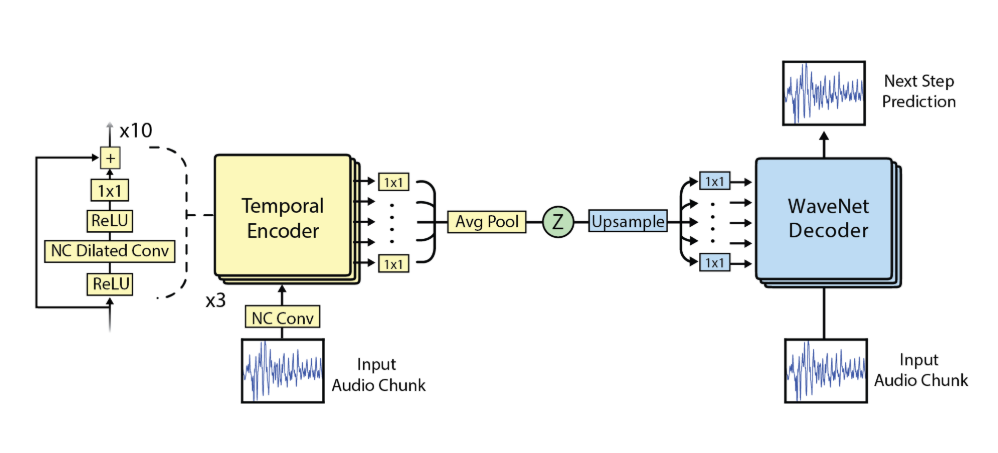
第二种选择：借助Google Colab 云平台，相当于Google在远程为我们提供了一台GPU性能优越的计算机供我们使用。

**二、实验原理**

Wavenet是一个在处理时间序列如语音和音乐信号方面效果显著的生成模型。首先，其核心概念是条件概率模型，简单地说，就是透过先前通过的信号信息去预测下一个点该有什么值。例如，对给定的输入序列(x\_1, x\_2, x\_3, x\_4, ..., x\_n) ， 每次要根据之前的x\_1 ~ x\_n来预测x\_n+1。然后将x\_n+1添加在输入序列，再由x\_2 ~ x\_n+1得到 x\_n+2，以此类推。由此我们就可以由一个原始的序列(x\_1, x\_2, x\_3, x\_4, ..., x\_n) 作为输入，按此模型进行构建，生成任意长度的序列。具体过程如下所示：

此外，由于语音的采样率高，时域上对感知范围要求大，如果神经元感受野不够会导致生成的语言或音乐逻辑性不强，即相关性较差。因此，Wavenet在上述模型基础之上，使用了扩张卷积的方法予以改善，即根据扩张大小选择连接的节点。比如ditation=1的时候，第二层只会使用第t，t +2，t +4......这些点，这样随着卷积层的增加，覆盖范围会指数增加。如下图所示：

之前的研究已经证明，如果Wavenet结合语言嵌入层（Linguistic Embeddings），能够在在语音文字转换方面取得令人满意的结果。但是Wavenet也存在着自己的缺点：一是运行速度受到较大限制，二是在生成较长序列时需要一段外部序列进行引导。

在本实验中，代码著作者通过使用自动编码器来学习它自己的嵌入层，实现降维与特征提取的效果，从而消除了对外部特性的限制。具体结构如下：

**三、实验代码**

**1.配置环境**

# 安装 magenta

print('Installing Magenta...\n')

!pip install -qU magenta

print('Installing ffmpeg...\n')

!echo "Yes" | apt-get install ffmpeg > /dev/null

print('Downloading Pretrained Models...\n')

print('Getting Instruments Model...\n')

!gsutil -q -m cp -R gs://download.magenta.tensorflow.org/models/nsynth/wavenet-ckpt.tar /content/

print('Getting Voices Model...\n')

!gsutil -q -m cp -R gs://download.magenta.tensorflow.org/models/nsynth/wavenet-voice-ckpt.tar.gz /content/

!cd /content/

!tar -xvf wavenet-ckpt.tar > /dev/null

!tar -xvf wavenet-voice-ckpt.tar.gz > /dev/null

print('Importing Modules...\n')

# 导入相关模块和辅助函数

import os

import numpy as np

import matplotlib.pyplot as plt

from IPython.display import Audio

%matplotlib inline

from google.colab import files

from magenta.models.nsynth import utils

from magenta.models.nsynth.wavenet import fastgen

from magenta.music.notebook\_utils import colab\_play as play

def upload(sample\_length, sr):

  '''Upload a .wav file.'''

  filemap = files.upload()

  file\_list, audio\_list = [], []

  for key, value in filemap.iteritems():

    fname = os.path.join('/content/', key)

    with open(fname, 'w') as f:

      f.write(value)

    audio = utils.load\_audio(fname, sample\_length=sample\_length, sr=sr)

    file\_list.append(fname)

    audio\_list.append(audio)

  return file\_list, audio\_list

download = files.download

get\_name = lambda f: os.path.splitext(os.path.basename(f))[0]

print('Sucess!! Environment is now setup.')

**2.选择训练模型（基于语音还是基于乐器音）**

Model = "Instruments" #@param ["Instruments", "Voices"] {type:"string"}

ckpts = {'Instruments': '/content/wavenet-ckpt/model.ckpt-200000',

         'Voices': '/content/wavenet-voice-ckpt/model.ckpt-200000'}

ckpt\_path = ckpts[Model]

print('Using model pretrained on %s.' % Model)

**3.导入待合成的音频**

#设定截取音频的长度（处理1s音频需要四分钟时间，不建议截取音频时间过长）

Length = 4.0 #@param {type:"number"}

SR = 16000

SAMPLE\_LENGTH = int(SR \* Length)

#上传音频（音频格式需为wav或mp3）

try:

  file\_list, audio\_list = upload(sample\_length=SAMPLE\_LENGTH, sr=SR)

  names = [get\_name(f) for f in file\_list]

  # 归一化处理

  for i in range(len(audio\_list)):

    audio\_list[i] = audio\_list[i] / np.abs(audio\_list[i]).max()

    if len(audio\_list[i]) < SAMPLE\_LENGTH:

      padding = SAMPLE\_LENGTH - len(audio\_list[i])

      audio\_list[i] = np.pad(audio\_list[i], (0, padding), 'constant')

  audio\_list = np.array(audio\_list)

except Exception as e:

  print('Upload Cancelled')

  print(e)

**4.编码、压缩**

audio = np.array(audio\_list)

z = fastgen.encode(audio, ckpt\_path, SAMPLE\_LENGTH)

print('Encoded %d files' % z.shape[0])

z\_list = [z\_ for z\_ in z]

name\_list = ['recon\_' + name\_ for name\_ in names]

n = len(names)

for i in range(n - 1):

  for j in range(i + 1, n):

    new\_z = (z[i] + z[j]) / 2.0

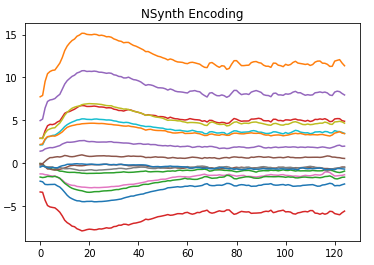
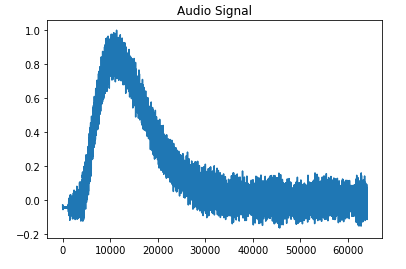
    new\_name = 'interp\_' + names[i] + '\_X\_'+ names[j]

    z\_list.append(new\_z)

    name\_list.append(new\_name)

print("%d total: %d reconstructions and %d interpolations" % (len(name\_list), n, len(name\_list) - n))

如下图所示，经过自编码器处理后的语音信号提取了原信号大部分结构特征。

****

**5.合成**

print('Total Iterations to Complete: %d\n' % SAMPLE\_LENGTH)

encodings = np.array(z\_list)

save\_paths = ['/content/' + name + '.wav' for name in name\_list]

fastgen.synthesize(encodings,

                   save\_paths=save\_paths,

                   checkpoint\_path=ckpt\_path,

                   samples\_per\_save=int(SAMPLE\_LENGTH / 10))

**6.播放合成音乐及原音乐**

print("Originals:\n")

for fname in file\_list:

  synth\_audio = utils.load\_audio(fname,

                                 sample\_length=SAMPLE\_LENGTH,

                                 sr=SR)

  print(get\_name(fname))

  play(synth\_audio, sample\_rate=SR)

for i, fname in enumerate(save\_paths):

  if i == 0:

    print("Reconstructions:\n")

  if i == len(file\_list):

    print("Interpolations:\n")

  synth\_audio = utils.load\_audio(fname,

                                 sample\_length=SAMPLE\_LENGTH,

                                 sr=SR)

  print(get\_name(fname))

  play(synth\_audio, sample\_rate=SR)

**四、运行结果**

运行结果以音频形式输出，已附在文件夹内。

**五、实验体会**

通过这次实验，我充分感受到了机器学习、神经网络蕴含的魅力，同时也意识到了自己在这方面仅仅是了解了一点皮毛知识，因为机器学习方面的知识、模型、算法简直是浩如烟海，其已经深入我们生活的方方面面，比如机器翻译、图像处理，甚至在新冠肺炎的鉴别方面深度学习都发挥了作用，深度学习通过从新冠肺炎患者的X光片中提取特征，进而实现新冠肺炎的自动初步判别，这一切都说明机器学习已和我们的生活密不可分了。在机器学习领域，Python是最常用的开发软件，因为其免费开源、且能搭载许多机器学习需要的模块，如Pytorch、tensorflow等等，所以说学好Python语言可以说是开始机器学习的第一步。除了软件，机器学习还需要强大的硬件支持，因为机器学习运算量大、复杂度高，因此一台GPU性能优越的计算机是先决条件。具体到这个实验，我接触了一种生成模型Wavenet，了解了它运行的基本原理，知道了扩张卷积在自然语言处理中的作用，它能使神经元获得更大范围的感受野，进而实现结合上下文关系处理语音、音乐信号的效果。例如，利用这个网络，我们可以生成介于吉他和笛子之间的乐器声，而这种乐器声，此前在自然界中是不存在的，这样我们就为音乐创作者提供了更广阔的发展空间。