**利用神经网络检测音乐BPM**

我一直想知道，使用基于神经网络的方法来检测一段音乐的拍子(或每分钟跳动的次数)是否可能。经过一段时间的实验后，我提出了采用循环神经网络（CNN）的解决方案。（**这是网上的一篇文章，其中有一些想法我认为是值得借鉴的，对于使用LSTM作为检测手段有指导作用**）

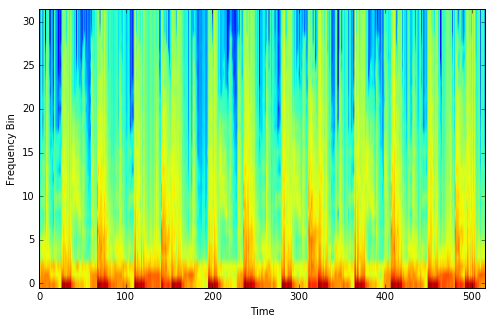
**方法**

首先，我提出一些关于表示输入音频、BPM和什么是理想的神经网络体系结构的最佳方法的想法。

**输入数据格式**

第一个想法是网络的输入应该采取什么样的一般形式。我不熟悉很多关于音频的物理结构，或者更广泛地说频率数据，但是我很熟悉[傅里叶分析](https://en.wikipedia.org/wiki/Fourier_analysis)和spectograms（频谱图）。

我认为一个频率spectogram可以作为对我正在计划训练的任何网络的适当输入。这些基本上包含了x轴上的时间，y轴上的频率。然后，值(像素颜色)指示音频信号在每个频率和时间步上的强度。



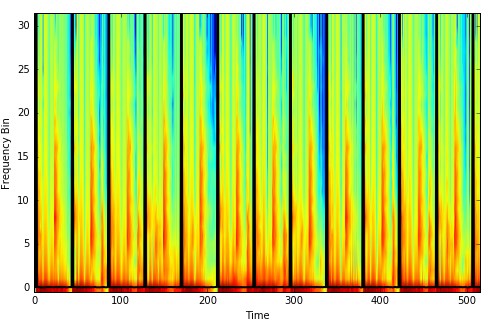
*一个短时电子音乐频谱图的例子，注意最低频中的鼓点数。*

**输出数据格式(由神经网络预测)**

我在这里有几个不同的想法。首先我想我可以直接预测BPM，然后我决定我可以通过尝试预测节拍的位置来保存网络的一些误差，以此推断BPM。我通过构造所谓的“脉冲向量”来实现这一点，如下:

* 我们有一个两秒的音频剪辑。我们可以用长度为200的零点的向量来表示这一点——每秒100帧的分辨率。
* 然后声明节拍是120BPM，第一次拍是在剪辑的开始。我们可以通过将这个向量的(零索引)元素[ 0，50，100，150 ]设置为1 (作为120BPM意味着每秒2次的节拍)来创建我们的目标向量。

我们可以相对容易地从这个向量推断BPM(尽管它的分辨率将决定如何准确)。作为回馈，网络也将(可能)告诉我们节拍的位置，除非他们经常出现。这可能是有用的，例如，如果我们想将两个轨道同步到一起。



*该图像覆盖目标输出脉冲矢量(黑色)超过音频剪辑的输入频率spectogram。*

**神经网络体系结构**

我的初始架构涉及到密集层。我使用Lasagne。在寻找一种方法将同一密度层应用于每一次的步骤时，我发现了keras的妙用。切换到keras后，我还添加了一个卷积层。因此，当前的结构本质上是一个卷积神经网络。下面进一步讨论了具体网络层的包含和顺序。

**创建培训数据**

主要的培训数据来自我的traktor收集。traktor是一个打碟程序，它可以检测你所给的轨道的BPM，特别是电子音乐。我没有安装traktor，但是我的音乐收藏中的很多MP3文件仍然有traktor-检测到的BPM存储在文件中。

我复制了大约30个MP3文件到一个文件夹，但是后来意识到他们仍然需要更多的审计文件，需要开始的第一拍，并且需要在假设BPM下的歌曲的整个歌曲超时。因此我打开每一个[收割机](https://nlml.github.io/neural-networks/detecting-bpm-neural-networks/reaper.fm" \t "_top)(一个数字音频工作站)，将每首歌曲剪辑成一首歌曲，确保他们没有超时，然后将其导出为wav。

**从MP3 /wav文件到培训数据都是由** mp3s\_to\_fft\_features.py **脚本**

我现在已经有了wav格式的歌曲，并且bpms从文件名中读取，进行手动输入。

然后将wav转换为spectogram。这是通过以下方式实现的:

1. 取长样本fft\_sample\_length(默认为768 )fft\_step\_size(默认为512 )
2. 在这些样品上执行快速傅里叶变换( FFT )

然后使用该函数创建目标脉冲矢量匹配wav的BPMget\_target\_vector.

然后随机子集长度desired\_X\_time\_dim分别从spectogram和目标脉冲矢量成对。通过这一点，我们生成大量的训练输入和输出，从仅仅一组训练输入到更易于管理的长度。每个样本代表大约6秒的音频，具有不同的偏移位置(因此我们的模型必须预测节拍的位置，以及它们发生的频率)。

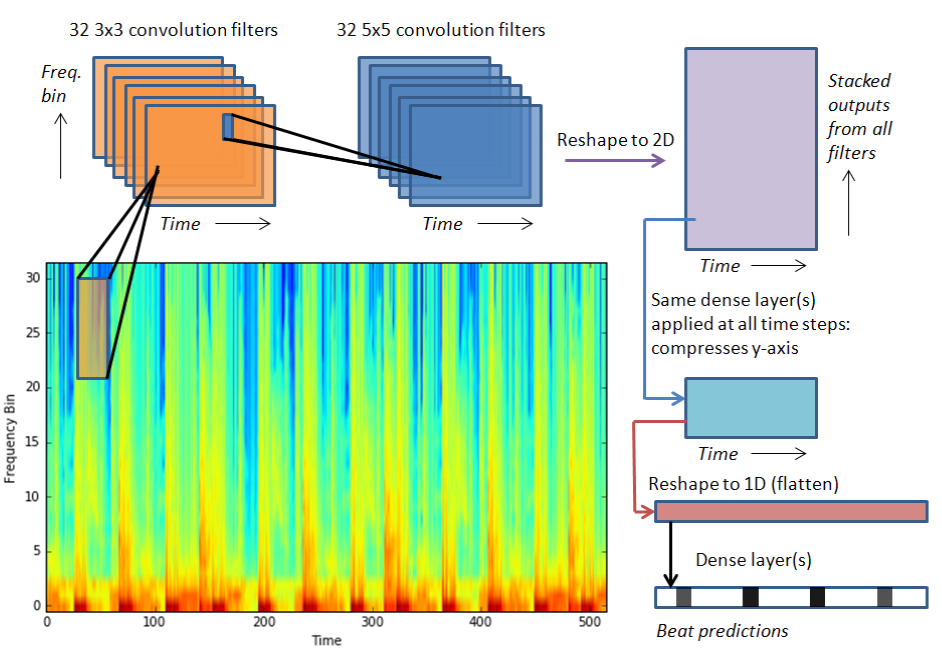
对于每一个～6秒的样本，我们现在有一个512x32矩阵作为训练输入-512时间框架和32频率箱(频率箱的数目可以通过增加downsample参数)-和一个512x1脉冲向量作为训练输出。

在最新版本的模型中，我有18首歌曲的样本。我通过前13首歌曲的抽样，以及从最后5首歌曲中的抽样来创建一个训练集。培训组包含28800个样本。

**指定和训练神经网络**

**网络体系结构.概述**

如上所述，我决定采用卷积神经网络结构。看起来像这样：



*神经网络体系结构概述。*

换句话说，图/体系结构可以描述如下:

* 输入spectogram通过两个顺序卷积层传递
* 然后，输出被重新编组为“由其他”表示的“时间”
* 然后使用keras的timedistributed密集层(在这些层中，每个时间步骤通过相同的密集层；这大大减少了估计所需参数的数目)
* 最后，输出被简化为一维，并在生成输出之前通过一些额外的desnse层

**网络体系结构-详细信息**

下面的代码片段给出了网络体系结构及其在keras中的实现的具体细节。

首先，我们有两个卷积层:

model = Sequential()

model.add(Convolution2D(num\_filters, 3, 3, border\_mode='same',

input\_shape=(1, input\_time\_dim, input\_freq\_dim)))

model.add(Activation('relu'))

model.add(MaxPooling2D(pool\_size=(1,2)))

model.add(Convolution2D(num\_filters, 5, 5, border\_mode='same'))

model.add(Activation('relu'))

我限制了最大池的数量。在第一个维度上的最大集合将减少时间粒度，我觉得这在我们的例子中很重要，在第二个维度中，我们没有多少粒度(只有32个频率分类器)。因此，我只在频率维度上执行了max池，并且只执行了一次。我仍然在试验卷积层的设置，但当前的configuartion似乎产生了良好的结果。

然后重新塑造卷积滤波器的输出，这样我们又有一个时间由其他事物的表示。这允许我们添加一些TimeDistributed层我们有一个类似512x1024的矩阵输入，有1024表示所有卷积的输出。这TimeDistributed层允许我们去像512x256这样的东西，但是只有一个( 1024x256)重量矩阵。然后在所有时间步骤使用此密集层。换句话说，这些层密集地将每个时间步骤的输出连接到在下面层的相应时间步骤中的输入。这一点的总体好处是需要学习的参数少得多。

这给我的感觉是，如果我们有一个代表每个时间步的1024长度的向量，那么我们可能会在那个时间步骤的较低维度中学习一个有用的表示，这将使我们达到一个矩阵大小，当我们尝试在之后添加一些密集层时，它实际上适合于内存。

model.add(Reshape((input\_time\_dim, input\_freq\_dim \* num\_filters)))

model.add(TimeDistributed(Dense(256)))

model.add(Activation('relu'))

model.add(TimeDistributed(Dense(8)))

model.add(Activation('relu'))

最后，我们平面化，添加几个稠密的层。这些同时考虑到时间和频率的维度。这应该是很重要的，因为模型可以尝试纳入实例，节拍应该是均匀地间隔时间。

model.add(Flatten())

for w in dense\_widths:

model.add(Dense(w))

model.add(Activation('relu'))

model.add(Dropout(drop\_hid))

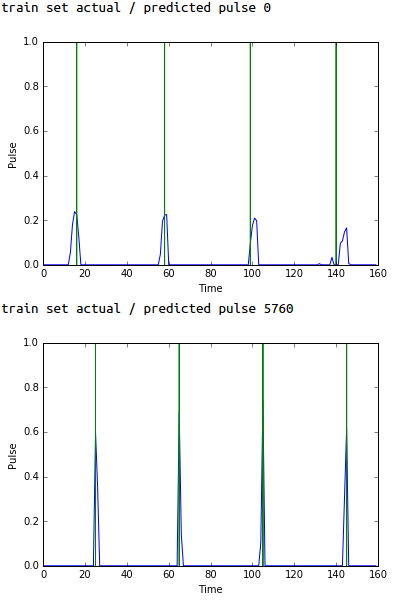
model.add(Dense(output\_length))

model.add(Activation('relu'))

**结果**

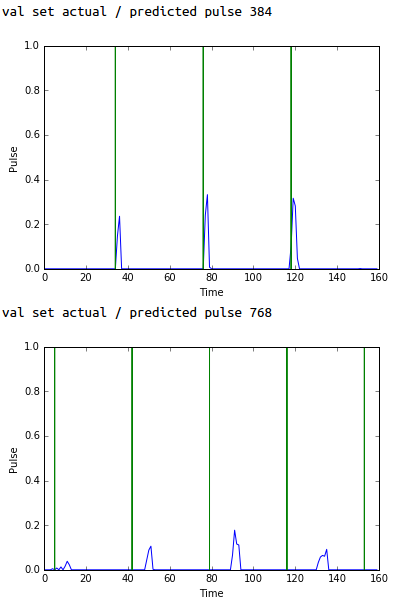
通常模型到运行到第九步时验证错误停止减少。

取决于当前的配置，模型能够在一定程度上检测到音乐中的节拍。注意，我实际上切换到了长度为160(时间维度)的输入和输出，尽管我能够在原始的512长度数据上获得类似的结果。

第一个图显示了培训组中的音频剪辑的典型性能:

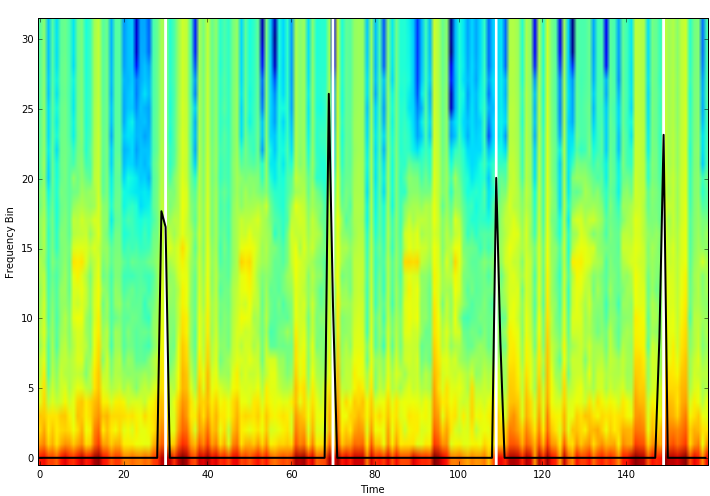
*预测(蓝色)与实际(绿色)脉冲-在训练集的典型性能。*

当试图预测来自不在训练数据中的歌曲的脉冲向量时，性能并不那么好。说，在一些歌曲，网络仍然可以发挥效用。它还经常得到正确的音乐节拍频率，即使这些节拍不在正确的位置:

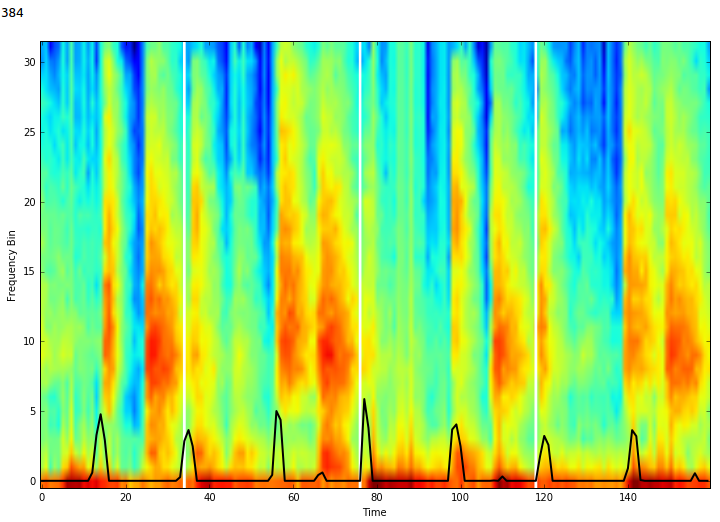


*预测(蓝色)与实际(绿色)脉冲-在验证集上的典型性能。*

如果我们通过输入训练数据绘制这些预测/实际值，我们可以将我们自己的直觉与神经网络的直觉进行比较:



*预测(黑色)与实际(白色)脉冲绘制在spectogram-典型性能超过培训集。*

以这个验证集为例。我会发现很难告诉你的节拍在哪里看着这个图像，但是神经网络至少能精确地测出它。

*预测(黑色)与实际(白色)脉冲在spectogram-典型性能上绘制的验证集。*

**下一步**

这仍然是一项正在进行的工作，但我认为，结果表明这种方法具有潜力。从这里我将继续探索:

* 使用更多的训练数据——我认为，神经网络需要更多的歌曲来学习指示音乐节拍的一般模式
* 我一直在考虑可能更好的方法:调整网络结构，在较短的音频块上执行“beat检测”，然后将这一时间的输出组合更长的时间。然后，这种更长的输出可以作为神经网络的输入，通过使用持续时间较长的上下文来“清理”beat预测