基于LSTM的计算机自动作曲

李旻骏

Abstract

音乐是人类最古老、最具普遍性和感染力的艺术形式之一，与此相应的，音乐作曲也被认为是需要大量知识积累的创作活动之一。能够让计算机完成这一任务有一定的意义以及挑战性。

在这篇文章中，我们尝试对于现有的利用人工智能进行计算机作曲的方法进行分析。效仿他们的工作，基于递归神经网络（RNN）与长短期记忆神经元（LSTM）理论，建立了一个自动作曲人工神经网络，并对此网络的一些有趣的输出进行了简单评价。

关键字：**计算机作曲，递归神经网络（RNN），长短期记忆神经元（LSTM）**

> [The Analytical Engine] might act upon other things besides number, ... the engine might compose elaborate and scientific pieces of music of any degree of complexity or extent.
>
> ——Ada Lovelace

1 介绍

想到作曲，人们通常都认为是一个需要创造力与高度智慧的工作，并认为仅有人类才有能力进行这种工作。的确，一部好的乐曲需要的东西是人们琢磨不透的，甚至可以说，作曲家也有对着空乐谱卡壳的时候。但这是不是意味着计算机不可能进行自动生成乐曲？我们如果把问题作为一种序列预测问题，那么这种问题就可以利用现有的计算机理论进行解决。

1.1 计算机作曲的意义

首先，我们认为这个问题是有意义的。

如果说音乐是人类的一种共通语言的话，那么音乐理论就是这种语言的语法规范。音乐理论的建立将古老、无序、难以名状的音乐带入了理论化、体系化的阶段。然而，在音乐充斥于人们日常的同时，与其相应的音乐理论却仍然因其理论性、教授方式等原因被束之高阁。乐理只能作为音乐专业人员的理论指导与分析工具，普通人群无法通过简单学习的方式将其的创意转化为音乐，利用计算机帮助人们作曲，可以使更多人有能力进行音乐的创造。

1.2 计算机作曲的挑战性

与此同时，作曲这本身就是一个具有挑战性的课题。

音乐有其简单的一面，如Fig.1，巴赫著名的平均律乐曲集中的第一首C大调前奏曲，这首乐曲仅由不间断地琶音构成，如开头$C \to D_m \to G^7 \to C$的和弦。虽然相比于平均律曲集中的其他曲目是过于简单，但是并不影响其的优美。又如Fig.2，贝多芬的献给爱丽丝，这首脍炙人口的曲子可以说是贝多芬最

简单的曲子之一，显示着简单的优美。不过与其相对，复杂的音乐其实占据大部分的古典乐曲的范畴，如Fig.3，同样是贝多芬的作品，开头就以重板和断奏压倒听众，随之而来的是快速下行音阶，然后再一次强烈的演奏。因为复杂的音乐有着更多的演奏工具与技巧，从而允许作曲家更加自由地表达自己的创作思想。

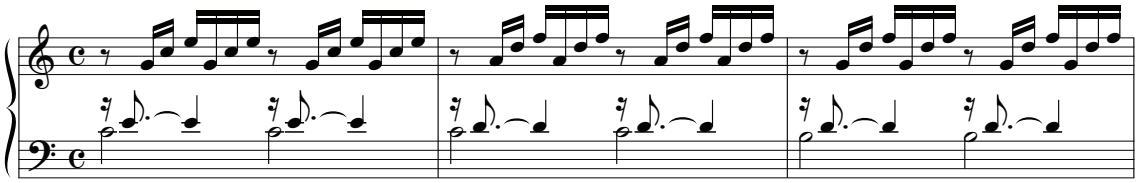

Figure 1: The Well-Tempered Clavier, Prelude in C major, BWV 846 (Johann Sebastian Bach)

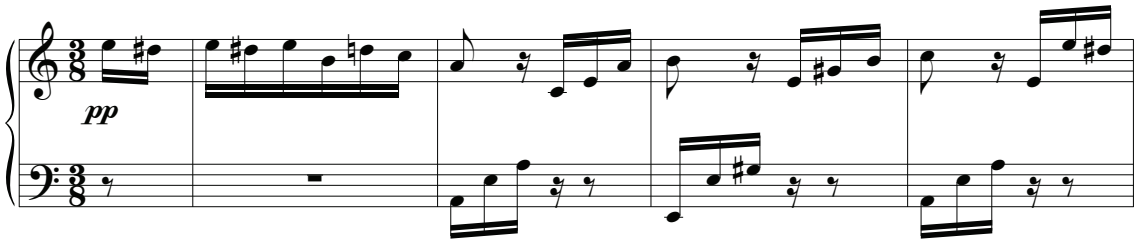

Figure 2: Fur Elise, WoO 59 (Ludwig van Beethoven)

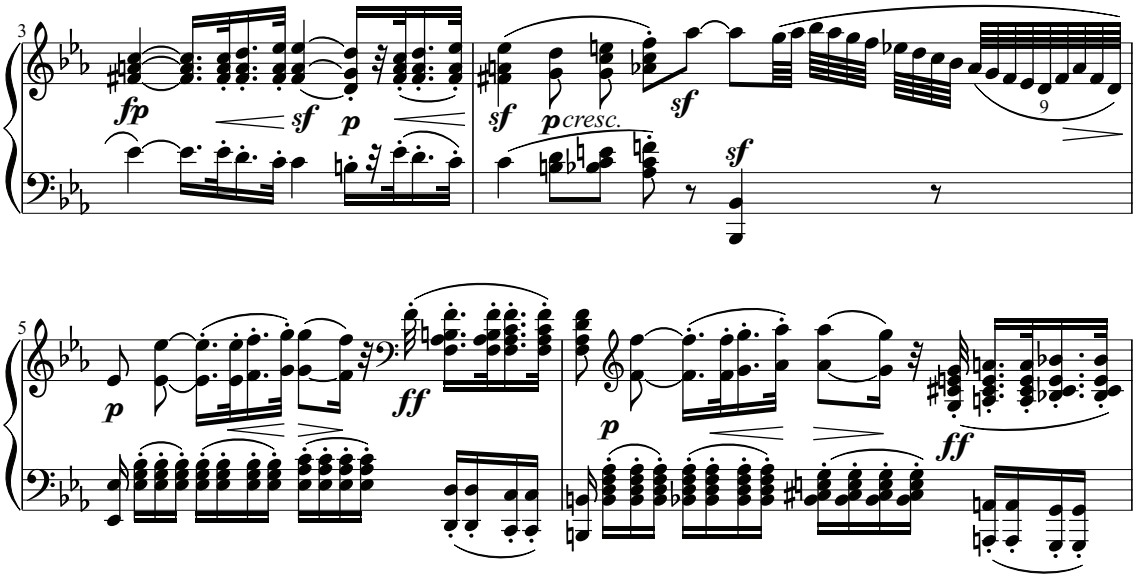

Figure 3: Piano Sonata No.8 in C minor, Op.13 ("Patheique") (Ludwig van Beethoven)

除了乐曲本身的复杂性之外，音乐背后有着各种复杂的性质与理论，如：乐器组成、乐曲结构、和声、对位等等。并且，我们到现在还仅仅是讨论古典音乐，除此之外的音乐世界更加复杂，难以一概而论。而要让一个计算机系统能够完成如此之多的任务比较不现实。因此我们在这里进行一些限制以方便我们对问题形式化：

- 乐器仅限于钢琴；
- 仅考虑音符的音长与音高，即忽视如力度等其他性质；

- 对简短的、长度为几个小节乐句进行作曲。

1.3 本文结构安排

在第2节，本文将对于计算机自动作曲的相关工作作简单的介绍。之后的第3节中会介绍我们建立的基于LSTM的RNN训练网络，并简单介绍其实现原理。其后的第4节中，将会介绍基于该网络在各个作曲家的乐曲上的训练的结果与分析。最后在第5节中，会对本文的工作进行总结。

2 相关工作的说明

此处对有关用RNN进行计算机作曲的工作进行简单的说明，更加详细或者其他方法的介绍可以参考[1]、[2]

如果要进行乐曲生成进行序列预测比较直接的方法就是用一个RNN来做序列预测。即学习在时间t的状态为输入的情况下，预测时间$t+1$的状态。

于是就可以从一个输入（可以是在训练集上选取一个作为输入），然后产生出一个新的状态。由新生成的状态作为输入又可以产生另外的状态，由此迭代下去。这种方法在1989年Todd就尝试过[3]

虽然RNN能够做到普通前馈网络不能做到的事情——记忆，但是实际上RNN并没有完全解决这个问题，因为它的“记忆”并不能保持长久。Mozer在1994年曾经对RNN音乐评论道“在小范围的乐句中的结果可以认同，但是这种音乐没有音乐的连贯性，没有乐曲的结构，以及只有一点的乐句结构。”[4] 缺少连贯性的原因是梯度（误差）在传播的过程中的衰减，导致不能实现长期的记忆。

于是LSTM就被提出来[5]解决这个缺点。粗略地说，其通过保护错误流使其不受时间影响，从而获得稳定的梯度（误差）值。

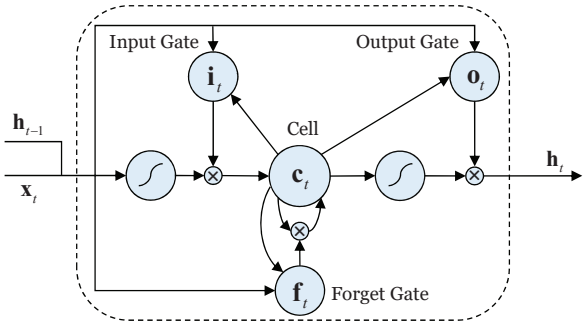

Figure 4: LSTM单元

其中$\{\mathbf{x}_i\}$为预测序列，$\mathbf{W}_{\alpha\beta}$为α与β间的权值。时间$t=1,2,\cdots$，$\sigma(x)$为sigmoid函数$\frac{1}{1+e^{-x}}$

$$
\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_{xi}\mathbf{x}_t + \mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{W}_{ci}\mathbf{c}_{t-1} + \mathbf{b}_i) \\
\mathbf{f}_t &= \sigma(\mathbf{W}_{xf}\mathbf{x}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{W}_{cf}\mathbf{c}_{t-1} + \mathbf{b}_f) \\
\mathbf{c}_t &= \mathbf{f}_t\mathbf{c}_{t-1} + \mathbf{i}_t \tanh(\mathbf{W}_{xc}\mathbf{x}_t + \mathbf{W}_{hc}\mathbf{h}_{t-1} + \mathbf{b}_c) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_{xo}\mathbf{x}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{W}_{co}\mathbf{c}_{t-1} + \mathbf{b}_o) \\
\mathbf{h}_t &= \mathbf{o}_t \tanh(\mathbf{c}_t)
\end{aligned}
$$

其在很多的序列预测问题中都展现了很强的的适应性[6]。

Eck尝试用基于LSTM方法来进行蓝调创作[1]，通过固定一组和弦，其网络可以输出一个音符在特定时刻被弹奏的概率。不过这种输出并不能表示一个音符被按住，一个常见的情况，这需要至少两个输出来表达。Boulanger-Lewandowski应用了RNN与限制玻尔兹曼机（RBM）来对复调音乐进行建模[7]。其中RNN负责处理时间的信息，RBM则负责产生对应的联想音符。这种方式感觉简洁，但是在实际听了输出之后感觉其并不能有效产生出好听的乐曲片段（可以参考`http://deeplearning.net/tutorial/rnnrbm.html`）。

3 方法

本文基于LSTM人工神经元构建RNN音乐建模网络，建立了一个如Fig.5所示的网络。最底层为输入层，为每个时间t的乐曲的状态$\mathbf{x}_t$，最顶层是输出层，链接着sigmoid函数。

在两者之间是两种LSTM网络。分别是时序网络与音符网络。我们这么做是希望网络首先能够理解音乐的次序，或者说时序性；同时还可以理解音符之间的关联性，或者说乐理。

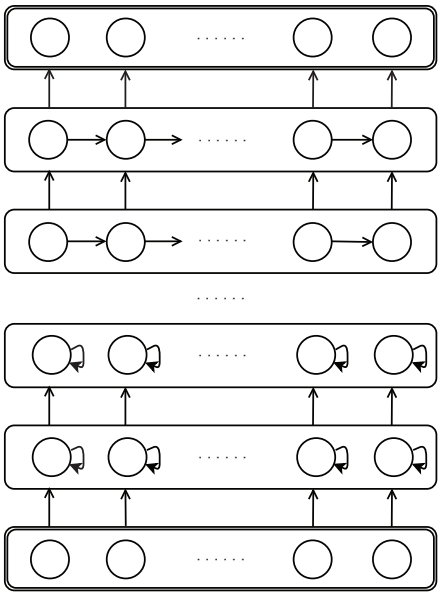

Figure 5: 乐曲学习网络

其中参考了[8]与[7]的工作，并参考了[8]的源代码进行构建。具体的实验代码参考src目录中的源码。运行源码需要安装依赖包Theano，Theano_lstm，python_midi，并且建议用GPU进行模型训练。

3.1 特征提取

3.1.1 MIDI格式读取与输出

整个系统对于音乐的输入输出是基于MIDI格式的音乐的，关于MIDI格式的内容，可以参考[9]。

MIDI格式中音符是以开始事件与结束事件来标记的，所以乐曲中音符可以由连续的线段表示，如Fig.6。

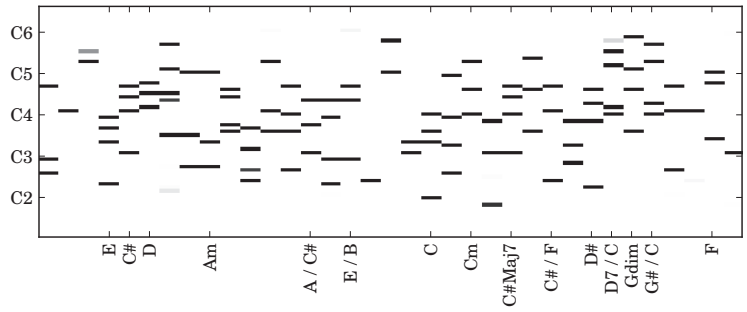

Figure 6: 一段MIDI格式的音乐[7]

在上文所提到过的，我们关心的仅有音符的音高（pitch）与时值（duration）。因此我们仅需从MIDI格式中获得以下信息：

MIDI格式中对于我们有用的信息（MIDI Event）：

- 音符按下事件
- 音符提起事件
- 乐曲的节拍

另由于如果同时有不同节拍的乐曲在一起训练的话，很容易产生不和谐音乐。因此我们在这里进一步追加约束：乐曲要是4拍子的。（因为4拍子的乐曲较为常见）

在我们提取了以上的事件之后，我们就可以定义乐曲在时间t的状态向量：

$$\mathbf{x}_t = \{x_{ti} \in enum\{0,1,2\} | i \in [minpitch, maxpitch]\}$$

其中x_{ti}可以取三个值$0,1,2$分别表示：无按下（松开），持续，按下，$[minpitch, maxpitch]$表示音高的取值范围。于是在这一时间的每一个音高都可以被这个向量描述。其中时间t的单位可以定义为$\frac{1}{32}$拍，这样可以有一定精度的对乐曲进行取样。

那么乐曲就可以表达为乐曲状态向量的序列（矩阵）：

$$\mathbf{M} = \{\mathbf{x}_t, t_s | t = 1, 2, \cdots, T\}$$

其中t_s是指拍号（Time Signatures），T为乐曲最大时间（$\frac{1}{32}$拍单位）。

3.1.2 特征选取

在获得了以上信息之后，知道了乐谱的最基本的信息。但是还需要进一步的特征提取之后才能作为网络的输入。

首先，网络的输入神经元是对应每个音符，因此我们应该以单一音符的角度思考。通过结合一些基础的乐理知识，我们可以得出以下一些比较重要的信息：

乐谱$\mathbf{M}$中重要的信息，从单一音符x_{ti}的角度：

- 音高
- 音符的唱名（A-G，12种）

- 音符附近的音符的状态（如一个八度的范围之内x_{ti}的值）
- 所有音符的状态（按照唱名压缩，如有多少个A被按下了）
- 音符处于小节中的位置

在乐理上，利用附近音符的状态可以得知这个音符起到了什么样的和声作用。例如：如果一个音相隔8个半音有另一个音符被激活，那么可以知道这两个组成平行六度，从而推断出其和声功能。

而利用所有音符的状态则可以得出现在进行的和弦。例如：如果C F G 都被按下的话，现在的和弦进行很可能是C大三和弦。

还有，由音符处于小节中的位置可以推断出该音符的处于强弱拍，然后得出这个音的性质，例如是否是切分音。

形式化上面的特征表示：

- 音高$x_{ti} \in [minpitch, maxpitch]$
- 音符的唱名$b_{ti} \in enum\{C, C^{\#}, D, \cdots\}$
- 音符附近的音符的状态$\mathbf{c}_{ti} = \{x_{tj} | j \in [i-12, i+12]\}$
- 所有音符的状态$\mathbf{d}_{ti} = \{\#(b_{tj} = b) | j \in [minpitch, maxpitch], b \in \{C, C^{\#}, D, \cdots\}\}$
- 音符处于小节中的位置$e_{ti} \in enum\{0, 1, \cdots, 15\}$

但是以上也仅仅是我们由乐理的角度得出，“如果是我的话我能够做出推断”。并没有确切的理由认为RNN也能利用这些材料做出推断，这需要进一步实验。

3.2 网络构建与训练

3.2.1 网络构建

对于以上得到的输入序列的特征状态向量

$$\{\mathbf{x}_{ti} = \{\mathbf{x}_t, x_{ti}, b_{ti}, \mathbf{c}_{ti}, \mathbf{d}_{ti}, e_{ti}\}\}, i \in [minpitch, maxpitch], t = 1, 2, \cdots, T,$$

我们可以把其作为LSTM网络的输入。

上面提到过的有两层网络分别适用于时序信息与音符信息。这样构建的原因是，如果仅仅有时序的信息流，那么一个音符的信息就仅能被这一LSTM序列学习。但其实乐曲不仅有着时间上的连续性，还有与周围音符的关联，或者说这是一种乐理的连续性。因此需要加上这样的关联使得保证乐曲在乐理上是连续的。

在第一类LSTM层（时序层）中，时序上LSTM神经元传递的是音符的特征状态向量；在第二类LSTM层（乐理层）中，低音高的LSTM神经元想高位神经元传递特征状态向量以及一个概率向量$\mathbf{y}_{tki}$，k标识神经元的序号。

$$\mathbf{y}_{tki} = (p(x_{(t+1)i} = 1), p(x_{(t+1)i} = 2))$$

就是说其还传递关于下一个时间这个音符的状态的概率。

最终网络输出

$$\mathbf{y}_t = \{\mathbf{y}_{ti} | i \in [minpitch, maxpitch]\}$$

3.2.2 训练算法

通过得到输出$\mathbf{y}_t$，我们可以把其与$\mathbf{x}_t$比较，我们用交叉熵$H(\mathbf{x}_t, \mathbf{y}_t)$来衡量两个概率分布的相似度。可以得到：

$$H(\mathbf{x}_t, \mathbf{y}_t) = -\sum_t \mathbf{x}_t \log \mathbf{y}_t = H(\mathbf{x}_t) - \sum_t \mathbf{x}_t \log \frac{\mathbf{x}_t}{\mathbf{y}_t}$$

其中和式后者就是两者的KL距离$D_{KL}(\mathbf{x}_t||\mathbf{y}_t)$。易得当两者相等时，$D_{KL}(\mathbf{x}_t||\mathbf{x}_t) = \sum \mathbf{x}_t \log(1) = 0$。又由于前者$H(\mathbf{x}_t)$为定值，因此我们可以得到当交叉熵最小的时候，$\mathbf{x}_t$与$\mathbf{y}_t$最相似（其实相等），即最理想的结果是两者相等。

但是相等并不好，因为这表示模型过拟合，我们的出来的结果就仅仅是一个复制品。所以我们要设置一定的dropout率，即在训练的时候随机屏蔽一些隐藏节点，防止节点过分相似。

具体的最做优化利用了Theano中已经实现的Adadelta梯度下降法，选择其是由于其有较稳定的性能。详细算法对比可以参考Karpathy在MNIST数据集上的比较`http://cs.stanford.edu/people/karpathy/convnetjs/demo/trainers.html`。

3.2.3 训练集

训练集的选择虽然并没有固定的方法，但是我们可以想到基本的要点是训练集内部不能有太大的差距。但同时我们也想到有很多的共性（如，和声、对位）都是古典音乐所共同的特征。因此我们首先打算通过一个“大古典音乐”数据集对模型进行基本训练，然后在对应各个特殊的古典音乐类别做Finetuning。

大古典音乐集

作曲家	乐曲（集）
Bach	The Well-Tempered Clavier, book 1 & 2
Bach	Inventions and Sinfonias
Bach	The Art of Fugue
Bach	The Goldberg Variations
Mozart	The Sonatas
Beethoven	The Sonatas
Chopin	Préludes, Opus 28
Haydn	The Sonatas
Brahms	Sonata C major, Opus 1

4 实验结果

本次实验在Amazon AWS 的$g2.2xlarge$ 实例上运行。耗时20 小时，完成10000 epoch 的基本训练。

4.1 混合古典音乐训练集的结果

由于失败的例子比较多，经过挑选之后得出的基本训练的较好的结果如Fig.7。

这一首曲子有四个声部，虽然其中有一个声部非常简单。由此可以看出这首曲子的巴洛克风格。没有出现训练集中有着的古典与浪漫乐曲的风格，出现这种情况可能是在训练集中巴赫的曲子太多了。

除此之外，由于很多生成的乐曲都受了巴赫的风格影响，都偏向于4声部或以上的复调音乐。但是由于复调音乐的理论性较强，本身就难以驾驭，丝毫的偏差都会影响整个乐曲的听感。

由此后面Finetuning都倾向于用简单的双声部或者非复调音乐作为训练集。

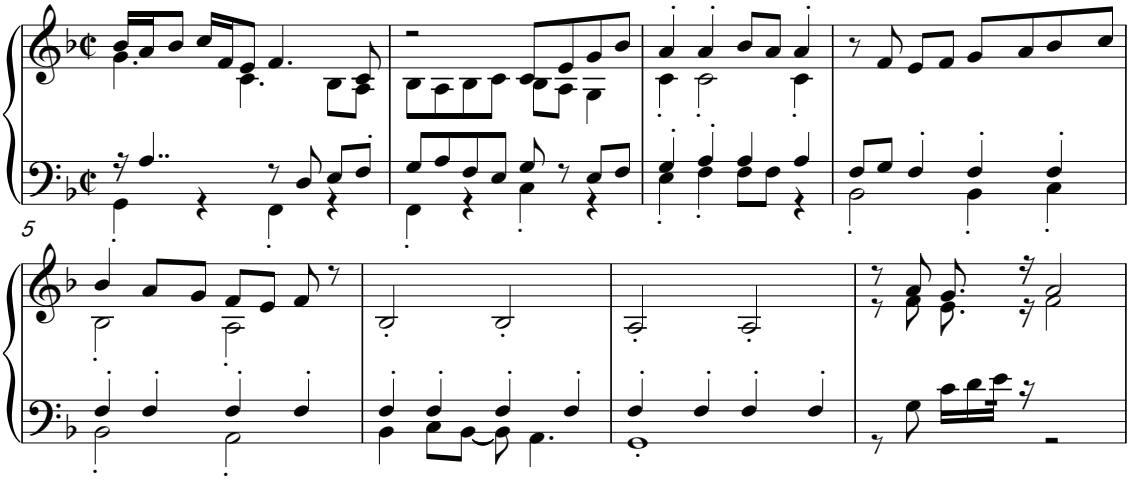

Figure 7: 训练后软件所作的古典乐曲（all.mid）

4.2 J.S. Bach （巴洛克时期）训练集的Finetuning

这里基于之前训练的古典乐曲10000 epoch 的模型继续训练，训练集用Bach 的Inventions 。经过1000 epoch 的训练后，得出结果如Fig.8。

由于训练集仅仅是双声部的乐曲，所以经过训练调整后输出的旋律都比较简单明了。仅仅是经过了1000 epoch ，但相比之前的模型，输出有了很大的改观。

并且，通过这个模型输出的结果失败的例子的比例相较于以前的模型有所减少。

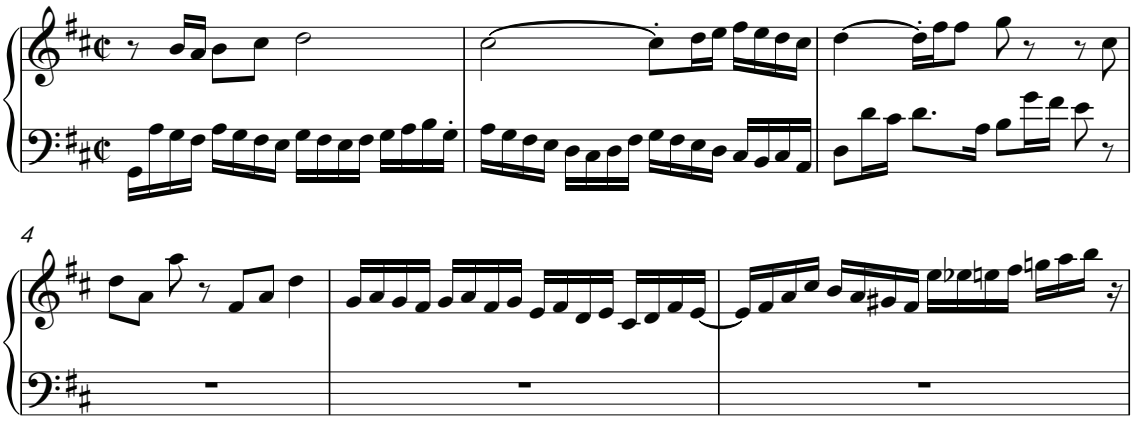

Figure 8: 训练后软件所作的巴赫风格的乐曲（bach .mid）

4.3 Mozart （古典时期）训练集的Finetuning

同样是基于之前训练的古典乐曲10000 epoch 的模型继续训练，训练集用Mozart 的Sonatas （所有，除了其中的第二乐章）。经过1000 epoch 的训练后，得出结果如Fig.9。

最终训练后生成的乐曲中有达到很高的相似度的曲子，如Fig.9中的。开头的第一小节明显就像着K.545第一乐章中的音阶上下行的旋律。整体的乐曲风格有着与莫扎特较高的相似度，可以说是最好的

结果。

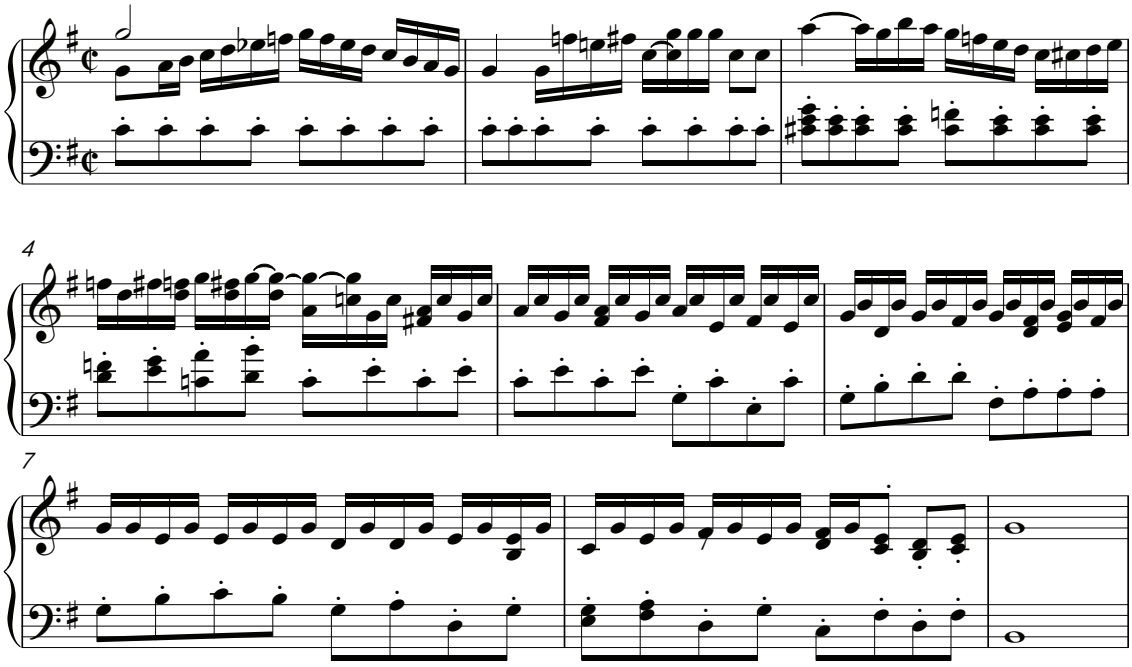

Figure 9: 训练后软件所作的莫扎特风格的乐曲（mozart.mid）

4.4 Chopin （浪漫主义时期）训练集的Finetuning

同样是基于之前训练的古典乐曲10000 epoch 的模型继续训练，训练集用Chopin 的Préludes, Opus 28 。经过1000 epoch 的训练后，得出结果如Fig.10。

这个结果算是在Chopin 训练的结果中比较能听的，但是节奏过慢，显得旋律不突出。

出现这种情况的原因可能是浪漫派的风格本身比较难用本文的方法学习，由于其长乐句多，和弦比较独特等等。使得之前基本上基于古典和巴洛克的模型比较难以转换过来。

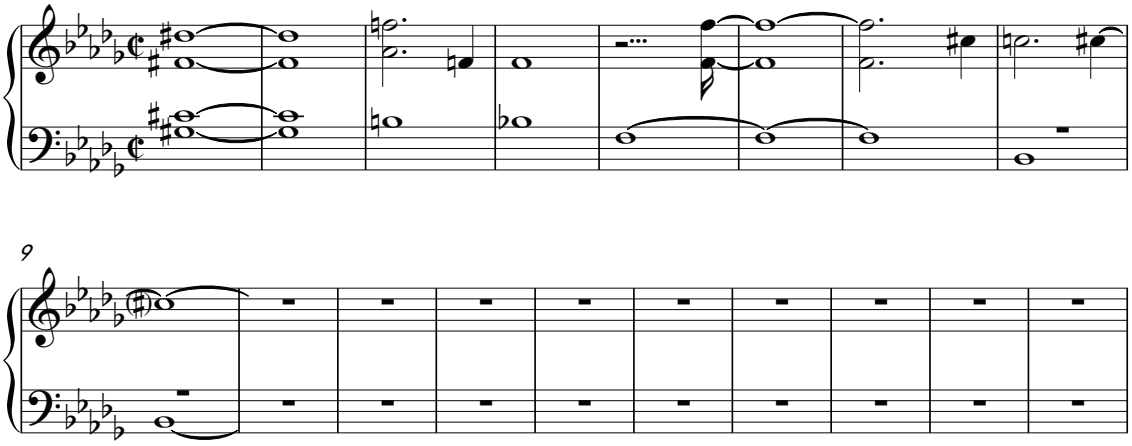

Figure 10: 训练后软件所作的肖邦风格的乐曲（chopin.mid）

4.5 小结

可以看到，我们的实现中用不同的训练集出来的结果有着很大的差距。原因主要是训练集的选择比较随意，没有斟酌数据集内部的一致性，导致原来的基础训练集偏向于巴洛克风格。最终使得整个模型的泛化能力较弱，如果基于更加复杂全面的训练集进行一次基本训练，相信效果会有所提升。

关于更多的输出，可以查看midi文件夹（作为对比，一些不好听的输出被放在failed文件夹中）。

5 总结

我们基于，通过用最简单的方法构造出了基于LSTM的自动作曲作曲的程序。基本达到了最初打算的计算机自动作曲的目的。

但其实我们的做法还有很多值得探索与改进的地方：如在生成的结果中其实能发现在一些结果之中有着原曲的影子，这样的究竟算不算是作曲？我们做的是不是一个拼凑？还有训练集应如何选择才能使得元模型有较好的泛化能力？RNN网络的结构应该怎么改进来更好地学习？这些问题都值得进一步探究。

Reference

[1] Douglas Eck and Juergen Schmidhuber. A first look at music composition using lstm recurrent neural networks. Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale, 2002.

[2] Alexis Kirke and Eduardo Reck Miranda. A survey of computer systems for expressive music performance. ACM Computing Surveys (CSUR), 42(1):3, 2009.

[3] Jamshed J Bharucha and Peter M Todd. Modeling the perception of tonal structure with neural nets. Computer Music Journal, pages 44–53, 1989.

[4] Michael C Mozer. Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multi-scale processing. Connection Science, 6(2-3):247–280, 1994.

[5] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997.

[6] Alex Graves. Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850, 2013.

[7] Nicolas Boulanger-Lewandowski, Pascal Vincent, and Yoshua Bengio. Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription. Proceedings of the 29th International Conference on Machine Learning (ICML-12), pages 1159–1166, 2012.

[8] Daniel Johnson. Composing music with recurrent neural networks. `http://www.hexahedria.com/2015/08/03/composing-music-with-recurrent-neural-networks/`, 2015. Accessed Jan. 10, 2016.

[9] Standard midi-file format spec. 1.1. `https://www.cs.cmu.edu/~music/cmsip/readings/Standard-MIDI-file-format-updated.pdf`. Accessed Jan. 10, 2016.