arXiv:2304.06868v1 [cs.SD] 14 Apr 2023

速度 VS. PI　TCH：理解自监督速度估计

*Gi*ovana Morais[1], *Matthew E. P. Davies*[2],*Marcelo Queiroz*[1],*Magdalena Fuentes*　[3]

[1] 计算机科学系，圣保罗大学，圣保罗，巴西[2] SiriusXM，美国
[3] MARL-IDM，纽约大学，纽约，美国

摘要

自监督方法通过解决无需人工生成标签的前置任务来学习表示，从而减轻了对耗时标注的需求。这些方法已应用于计算机视觉、自然语言处理、环境声音分析，以及近年来的音乐信息检索领域，例如音高估计。尤其是在音乐语境下，关于这些模型面对不同数据分布时的脆弱性，以及如何缓解这种脆弱性，目前仍缺乏深入认识。本文通过严谨的合成数据实验，剖析一个为音高估计设计并改用于速度估计的自监督模型，以探讨这些问题。具体而言，我们研究了输入表示与自监督速度估计的数据分布之间的关系。

索引词— 自监督、速度估计。

1. 引言

速度是音乐的一个基本维度，与听者为标记底层节拍而随音乐打拍子的速度有关。过去十年间，自动速度估计受到了广泛关注；由于给定节奏的节拍结构可能有多种解读，速度本身也存在固有歧义，再加上音符起始较弱、切分音以及局部速度变化等因素，速度估计至今仍被视为一项具有挑战性的任务 [1]。由于具有诸多潜在应用，速度估计在音乐信息检索（MIR）领域受到了相当多的关注[2, 3, 4], , 例如自动生成播放列表和推荐，或作为其他音乐信息检索任务（如转录或同步）的输入 [5]。正如音高是谐波声音的参照，速度在节奏语境中则是周期性的参照 [5]。

早期方法主要基于信号处理，将自相关或傅里叶分析等技术与峰值拾取相结合 [5, 6], ；近期方法则利用深度神经网络（DNN）。与其他音乐信息检索任务类似，深度神经网络的引入带来了范式转变，并提高了速度估计模型的准确率。探索卷积神经网络（CNN） [2], 以及更近来探索时序卷积网络 [4] 的新方法达到了最先进水平。

尽管深度神经网络显著提升了自动估计速度的效果，但众所周知，这类模型对数据的需求量很大 [4]。这在音乐信息检索领域尤为棘手，因为获取标注数据成本高昂，而制作标签往往需要音乐专业知识。在计算机视觉等其他领域，这一数据瓶颈促使人们采用自监督方法。这类方法通过算法从数据中生成标签，无需人工监督。在自监督学习中，模型经过训练以解决前置任务。这项任务可能与模型最终应用的下游任务无关，但会利用数据中的某些内在信息。若设计得当，模型在解决前置任务的过程中能够学到有意义的内部表征，并将其迁移到目标问题中。这种方法已应用于计算机视觉、环境声音分析，以及最近的音乐信息检索领域，用于音高 [7]和节拍跟踪 [8]。

深度神经网络的一大缺陷是，我们并不了解模型究竟在学习什么，也不清楚它们面对不同训练数据分布时有多脆弱；引入自监督也未能改善这一点。例如，研究表明，在自监督设置下，使用不平衡分布训练的模型会将这种偏差迁移到下游任务中，导致后续阶段难以纠正这一行为 [9]。因此，了解数据分布与模型之间的相互作用，有助于决定使用哪些数据进行训练，以及如何实施数据增强。

本文受音高估计与速度估计这两项任务之间相似性的启发，将一个用于音高估计的自监督模型 [7] (SPICE)改造用于速度估计，以此探讨这些问题。具体而言，我们利用合成数据考察输入表示与数据分布之间的相互作用及其对该模型性能的影响。随后，我们讨论这种方法的鲁棒性及其适应真实数据的能力。

2. 相关工作

Desblancs 等人 [8]研究了自监督节拍跟踪，他们训练了两个不同的卷积神经网络，一个处理信号的打击乐内容，另一个处理信号的谐波内容，并让网络完成同步性预测这一前置任务。尽管这项工作前景广阔，但它依赖计算成本高昂的声源分离来提取信号中的打击乐和谐波部分，从而延长了训练时间。

此前关于自监督音高估计的研究 SPICE [7],提出了估计相对音高的概念：将恒定 Q 变换（CQT）的两个移位片段输入同一个卷积自编码器进行拟合，并训练编码器，使其输出之差与所施加的移位成正比。这一前置任务取得了很好的效果，同时将预处理工作降至最低：只需计算恒定 Q 变换并进行移位。

与幅度频谱图或恒定Q变换表示信号中对音高分析至关重要的时频变化内容类似，速度图表示会针对每个时间帧，标示给定音乐录音中某个速度估计在局部的相关性 [10]。因此，将 SPICE 这类架构调整为以速度图作为输入、用于速度估计，是一个很有吸引力的方向

表示。基于此，我们探索了将 SPICE 调整为从速度图进行速度估计，并研究了不同的设计选择及其对速度估计的影响[1]。

3. 自监督速度估计

3.1. 速度表示

大多数速度图表示分两个阶段计算 [10]。首先计算新颖度函数，然后分析该函数的周期性，以获得局部速度显著性图。在此分析过程中，主要挑战之一与音乐分层节奏结构中存在的多种周期性有关，这些周期性在速度图中表现为谐波或次谐波。尽管这种“谐波”组织形式类似于频谱图表示中的谐波声音结构，但它可能更具歧义：一段音乐中占主导地位的感知速度在速度图中的显著性可能较低，而其谐波（或次谐波）的显著性却可能很高。这种“谐波结构”取决于音乐本身以及计算速度图所用的方法（例如，自相关或傅里叶分析）。

为了理解这些多种周期性对我们的自监督框架所产生的影响，我们比较了模型在采用三种速度图形式时的表现：自相关速度图 $\mathcal{T}_{\mathrm{A}}$ [12]、基于傅里叶的速度图 $\mathcal{T}_{\mathrm{F}}$ [13]，以及混合 速度图表示 $\mathcal{T}_{\mathrm{H}}$ [5]，后者结合了前两种形式。为此，我们使用单频带谱通量计算新颖度函数，并据此计算 $\mathcal{T}_{\mathrm{A}}$ 和 $\mathcal{T}_{\mathrm{F}}$。随后，我们将 $\mathcal{T}_{\mathrm{A}}$ 从时滞表示转换为时间-速度表示 [12]。由于速度值对应于线性采样的时滞值，此转换会产生非线性速度值。为了获得与 $\mathcal{T}_{\mathrm{F}}$具有相同速度集合的自相关速度图 $\mathcal{T}_{\mathrm{A}}$，我们对速度值 [10] 进行插值。最后，我们计算 $\mathcal{T}_{\mathrm{H}} = \mathcal{T}_{\mathrm{A}} \times \mathcal{T}_{\mathrm{F}}$。

频率变化带来的感知因频率范围不同而异，例如，从 40 Hz 增至 80 Hz，增加了 40 Hz，会被感知为一个八度；而从 120 Hz 增至 160 Hz，增加了 40 Hz，则会被感知为一个纯四度。同样，速度变化的感知也取决于其速度 [14, 15, 16],，因此从每分钟 60 拍增至每分钟 90 拍（每分钟节拍数，bpm），会被感知为比从每分钟 130 拍增至每分钟 160 拍更大的变化。考虑到这一点，我们将速度图表示中的速度轴从线性刻度转换为对数刻度，方法是将线性频格合并为以

$$t_k = t_0 2^{\frac{k}{Q}}, \tag{1}$$

每分钟节拍数为 $t_0 = 25$、每个八度有 $Q = 40$ 个频格的对数频格。这与从（线性频率）频谱图推导出恒定Q变换（其采用对数频率轴）的方式颇为相似。

3.2. 自监督框架

我们的自监督框架[2] （图 1）受 SPICE [7]启发。给定一张速度图，我们的方法会提取两个随机的 128 维切片 x_1 和 x_2，它们对应同一时刻，但分别在垂直方向偏移 k_1 和 k_2 个区间。这些偏移会使输入模型的不同切片呈现人为的速度变化

[1]在提交本文时，一篇相关且同期发表的论文 [11] 已于 2022 年 9 月 3 日在线发布。[2]我们的实现已开源，可在 https://github.com/giovana-morais/steme 获取

由于速度图的各行发生了垂直位移，这些切片会被输入模型。我们从均匀分布 $\mathcal{U}$（11，.....18）中采样数值 k_1 和 k_2，因此模型看到的速度范围为每分钟 30 拍至每分钟 310 拍；也就是说，对于公式 1 中的 $k = 11$，切片从每分钟 $t_{11} = 25 \cdot 2^{\frac{11}{40}} = 30.2$ 拍开始，到每分钟 273。2 拍结束；类似地，对于 $k = 18$，输入切片覆盖的区间为每分钟 $[34.1, 308.4]$ 拍。

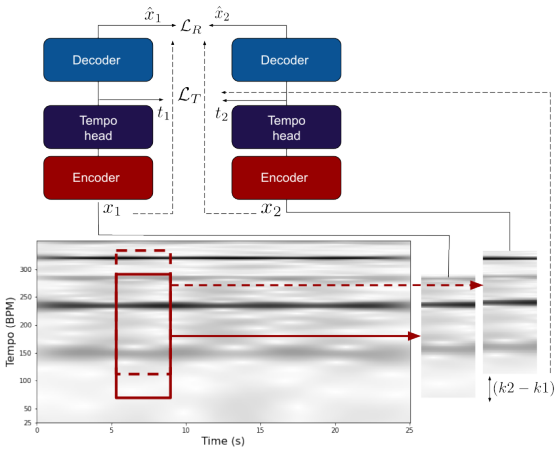

图 1。训练框架概览。

与 [7],一样，我们将相对速度误差定义为

$$e_t = |(t_1 - t_2) - \sigma(k_2 - k_1)|, \tag{2}$$

因此，各分支中估计的速度之差必须与人为引入的差值（$k_2 - k_1$）成正比。常数 σ 确保模型输出中的差异 $t_1 - t_2$对应于训练集中的正确区间数，其取值设为

$$\sigma = \frac{1}{Q \log_2\left(\frac{t_{\max}}{t_{\min}}\right)} \tag{3}$$

根据 [7],，其中 $t_{\max}$ 和 $t_{\min}$ 分别是训练集中速度的最大值和最小值，Q 是每个速度八度的区间数。我们将 $Q = 40$ 设为一个合理的速度范围和分辨率，例如，四个速度八度（30–240 每分钟节拍数）对应 160 个区间。与 [7], 不同，我们观察到，$k_2 - k_1$（而非 $k_1 - k_2$）保留了每分钟节拍数与模型输出值之间的单调对应关系，因为较大的偏移对应于速度图中显著性线较低、因而速度较低的切片。

随后，通过对速度误差应用 Huber 范数来训练模型，使其估计每个切片内的相对速度，从而得到以下损失函数：

$$\mathcal{L}_T = \frac{1}{T} \sum_t h(e_t), \tag{4}$$

其中 T 表示批量大小，h 定义为

$$h(x) = \begin{cases} \frac{x^2}{2}, & \text{if } |x| \le \delta, \\ \frac{\delta^2}{2} + \delta(|x| - \delta), & \text{otherwise}, \end{cases} \tag{5}$$

并且我们设定 $\delta = 0.25$。

除了公式 4 中的损失外，与 [7]类似，我们还使用如下重构损失：

$$\mathcal{L}_R = \frac{1}{T}\sum_T ||x_1 - \hat{x}_1||_2^2 + ||x_2 - \hat{x}_2||_2^2, \tag{6}$$

其中，x_1，x_2 是输入到各模型分支的输入切片，而$\hat{x}_1$，$\hat{x}_2$ 是图 1 中解码器输出的重建切片。最终的组合损失如下：

$$\mathcal{L} = \omega_T \mathcal{L}_T + \omega_R \mathcal{L}_R, \tag{7}$$

其中 $\omega_T = 10^4$ 和 $\omega_R = 1$（如 [7]所示）。

我们采用一个包含 6 层的卷积编码器，它接收一个 128 维向量（对应速度图的一个切片），并输出一个表示速度的标量。我们使用大小为 3、步幅为 2 的滤波器，通道数设为 $d \cdot [1, 2, 4, 8, 8, 8]$，其中 $d = 64$；所有层均使用 ReLU 作为激活函数。最后一个卷积层的输出会被展平，并输入速度估计头，该头由两个全连接层组成，单元数分别为 48 和 1。在训练过程中，我们会打乱输入音频的速度时频图帧，以确保每个批次都能看到不同的速度。随后，编码器的标量输出会输入解码器，解码器的目标是根据该速度估计重建速度图切片。解码器也包含 6 层（如 $d \cdot [8, 8, 8, 4, 2, 1]$），由转置卷积层组成，其滤波器、步幅值和激活函数均与编码器相同。网络两个分支之间共享权重。

与 [7], 不同，我们没有置信度／有声性头，也未使用最大池化或批归一化层，因为非正式测试表明，加入这些步骤不利于学习。

3.3. 模型校准

速度头的输出是一个实数标量 $t \in [0, 1]$，表示给定片段内的估计速度。要使用该模型进行推理，我们需要通过校准步骤 [7] 将此输出映射到 BPM 范围。考虑到速度图的对数特性，给定一个线性的每分钟节拍数值范围，模型输出的数值将呈对数分布，其中较高速度对应的数值会挤压到 t 中更为接近的激活值。基于此，我们按照对数速度范围生成合成节拍器点击声，并使用线性模型将模型输出映射到每分钟节拍数。

4. 实验

我们研究速度图输入表示的影响、数据分布的作用，以及二者在自监督速度估计中的相互作用。为此，本研究重点分析合成数据，并通过训练九个模型开展受控实验，分别对应数据分布与速度图的每一种组合。此外，我们还针对每种速度时频图表示，使用真实数据训练一个模型作为比较基准。

4.1. 合成数据生成

为了了解数据分布如何影响自监督模型性能，我们按照四种不同的分布合成节拍器片段（$N = 1000$）。其中三种将速度建模为对数正态随机变量 X，使得

$$\log_{0.25}(X) \sim \mathcal{N}(\mu, \sigma^2),$$

其中 $\mu \in \{70, 120, 170\}$ 每分钟节拍数，且 $\sigma =$。这三种分布参考真实世界的数据集（例如 GTZAN [17, 18]），并将其偏移至不同的速度范围。我们还纳入了一种对数均匀分布，其中 $\log_2(X) \sim \mathcal{U}([30, 240])$，代表一种均衡良好的理想速度分布。这种分布在实践中很难找到，但可作为我们实验的性能上限参考。结果分布见图 2。

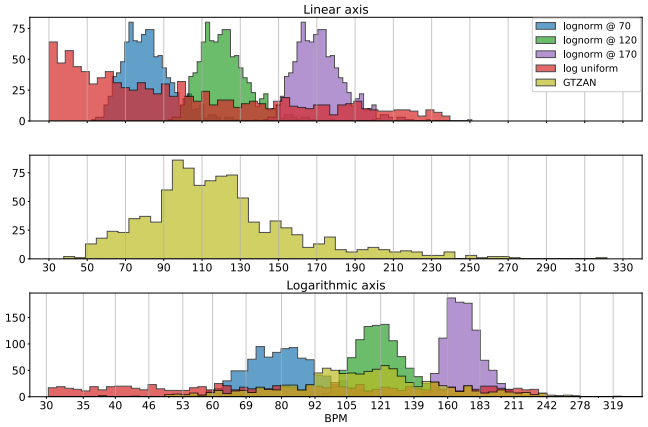

图 2。合成数据与 GTZAN 分布。第一行显示线性坐标轴上的合成数据分布。第二行显示线性坐标轴上的 GTZAN 数据分布。第三幅图显示对数坐标轴上的所有数据分布。可以看出，在对数坐标轴上（对应于速度图中的线条），以较高速度为中心的分布，其支持范围比以较低速度为中心的分布更窄。

4.2. 实现与评估

该模型使用 Keras [19] 实现，并使用 Adam [20] 和默认超参数进行训练，学习率设为 10^{-4}。批量大小为 64，所有实验均训练了 15 个训练轮次。所有速度图中使用的频谱通量，均通过 librosa [21], 计算；音频采样率为 22 kHz，窗口大小为 2048，跳跃长度为 512 个采样点。我们使用 $\gamma = 100$ 进行对数压缩，并对所得信号进行归一化。速度图采用 10 秒窗口，跳跃长度为 1 个频谱通量采样点（~ 23.2 毫秒）。速度图的实现基于 libfmp [22]。

所有合成数据均由 60 秒的节拍器音轨组成，这些音轨遵循不同的速度分布。对数正态分布使用 SciPy [23] 生成，而对数均匀分布定义为 $t_{\min} \cdot e^{\log(i \cdot t_{\text{最大值}}/t_{\text{最小值}})}$，其中 $i \in [0, 1)$。

4.3. 结果与讨论

由于校准结果与模型性能相关，即校准曲线越线性、离散程度越低的模型，其估计值越一致 [7], ，因此我们查看了各模型的校准结果，如图 3 所示。

傅里叶速度图在不同数据分布上呈现出更一致的校准结果，并且在合成数据和真实数据上都能产生更平滑的曲线。这一结果符合我们的直觉：傅里叶速度图与 [7] 中的常数Q变换表示最为相似，因为它往往包含上谐波而不包含次谐波。在

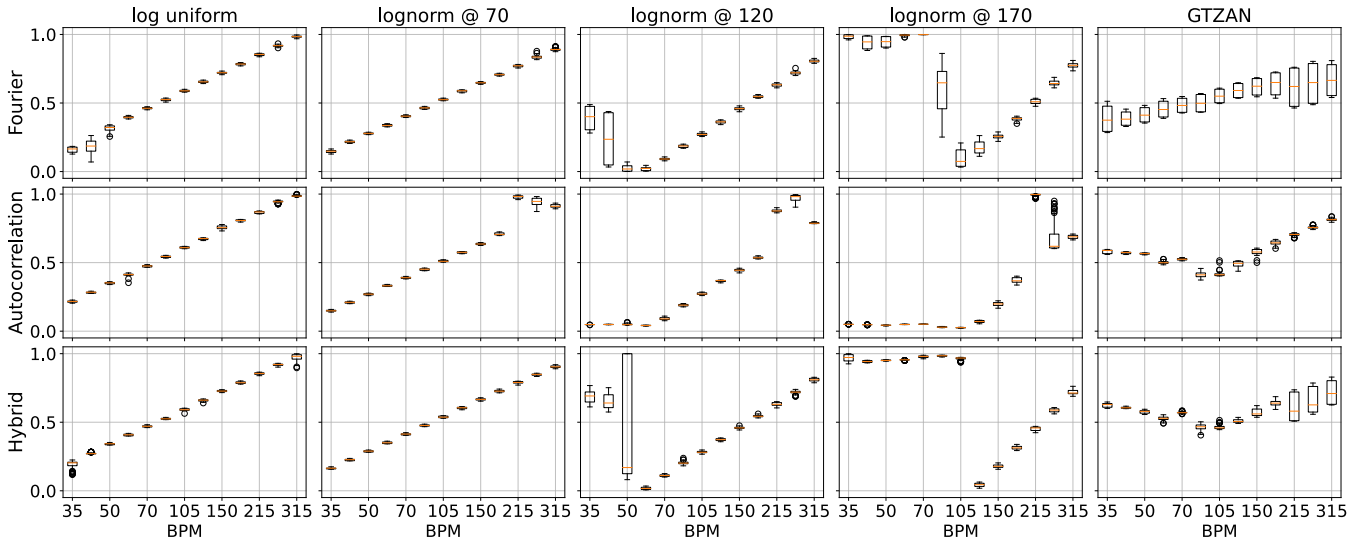

图 3。模型校准结果 适用于不同的数据分布和速度图表示。各图共用 x 轴和 y 轴 s.

相反，自相关速度图的结果变化最大，这意味着对于相近的速度值，模型难以给出相同的速度输出。令人意外的是，混合速度图的表现不如傅里叶速度图，这表明自相关速度图与傅里叶速度图相乘会去除模型进行速度估计所依赖的谐波信息。

针对不同数据分布进行的实验表明，模型并未从对数均匀分布更宽的范围中获益；相反，它能够根据以较低速度范围（70 每分钟节拍数）为中心的对数正态分布进行外推。仔细观察这一结果，可以从图 2 中找到解释。给定一个以线性每分钟节拍数值表示的速度分布（图 2 上图），由于输入速度图的对数效应，模型接收到的速度分布会略有不同（图 2 下图）。这种对数表示会随着速度升高而越来越压缩速度范围，这意味着模型实际可利用的对数每分钟节拍数值范围更小。在更极端的情况下，例如以 170 每分钟节拍数为中心的对数正态分布，模型会学着用整个输出范围来表示它所见到的较小速度值范围，并在范围外的数值处“饱和”，也就是说，它会为所有未见过的速度分配相同的值。这表明，在较低速度范围内标注或增强数据会更有益，从而确保模型更加稳定。

最后，观察使用真实数据训练的模型结果时，我们发现同样的结论依然成立。傅里叶速度图能够得到更稳定的模型；与使用合成数据训练的模型相比，该模型的方差更大，这是因为真实数据固有的可变性和噪声使训练更加复杂。GTZAN 分布与较低每分钟节拍数的对数正态分布相似，这一点体现在采用傅里叶速度时频图表示的图 3 中平滑的趋势上。对于 GTZA N 数据，混合速度图的表现也不如另外两种变体，这表明如果不对信号进行进一步处理就将两种速度图结合起来，可能会引入伪影和噪声；而对于合成数据，由于其简单且稳定，这样做已足够，如[5], 所示。

5. 结论

我们研究了自监督速度估计中速度图输入表示与数据分布之间的相互作用，并使用合成数据展示了将 SPICE 从音高估计改用于速度估计时面临的挑战和需要考虑的问题；在当前阶段，我们的目标并非直接推进速度估计领域的最先进水平。我们发现，傅里叶速度图能够得到更一致、更精确的模型；此外，当观测到的每分钟节拍数值像真实数据分布那样集中时，以较低速度值为中心的分布能够带来更成功的模型。未来，我们将结合真实数据进一步探索这些发现，并利用这些见解来指导数据增强。

6. 致谢

感谢 IEEE 信号处理学会和 ME-UYR 项目。第三位作者获得 CNPq 310141/2022- 2 号基金资助。我们还要感谢 F´elix Davies 富有成效的讨论。最后，感谢匿名审稿人对本研究提出的意见和反馈。

7. 参考文献

[1] Hendrik Schreiber、Juli´an Urbano 和 Meinard M¨uller，《音乐速度估计：我们已经完成了吗？》，载于国际音乐信息检索学会会刊，2020 年 8 月，第 3 卷，第 111–125 页。[2] Hendrik Schreiber 和 Meinard M¨uller，《使用卷积神经网络进行音乐速度估计的单步方法》，载于国际音乐信息检索学会会议（*ISMIR*），2018 年，第 98–105 页。[3] Hadrien Foroughmand 和 Geoffroy Peeters，《用于速度估计和节奏模式识别的 Deep-rhythm》，载于国际音乐信息检索学会会议（*ISMIR*），2019 年，第 636–643 页。

[4] Sebastian B¨ock 和 Matthew E P Davies，《解构、分析、重构：如何改进速度、节拍和强拍估计》，载于国际音乐信息检索学会会议（*ISMIR*），2020 年，第 574–582 页。[5] Geoffroy Peeters，《基于模板的时变速度估计》，*EURASIP*《信号处理进展期刊》，第 2007 卷，第 1–14 页，2006 年。[6] Jose R Zapata 和 Emilia G´omez，《音频速度估计方法的比较评估与组合》，音频工程学会期刊，第 198–207 页，2011 年。[7] Beat Gfeller、Christian Frank、Dominik Roblek、Matt Sharifi、Marco Tagliasacchi 和 Mihajlo Velimirovic，《SPICE：自监督音高估计》，*IEEE/ACM*《音频、语音与语言处理汇刊》，第 28 卷，第 1118–1128 页，2020 年。[8] Dorian Desblancs、Romain Hennequin 和 Vincent Lostanlen，《Zero-Note Samba：自监督节拍跟踪》，工作论文或预印本，2022 年 5 月。[9] Ho Hsiang Wu、Magdalena Fuentes 和 Juan P Bello，《探索用于音乐分类的模态无关表示》，载于第 *18* 届声音与音乐计算会议（*SMC 2021*），2021 年，第 191–198 页。[10] Meinard M¨uller，《音乐处理基础：音频、分析、算法与应用》，Springer Publishing Company, Incorporated，第一版，2015 年。[11] Elio Quinton，《用于音乐速度估计的等变自监督》，国际音乐信息检索学会会议（*ISMIR*），2022 年。[12]《FMP 笔记本：自相关速度图》，https: // www.audiolabs-erlangen.de/resources/MIR/ FMP/C6/C6S2 TempogramAutocorrelation.html，访问日期：2022-10-25。[13]《FMP 笔记本：傅里叶速度图》，https: // www.audiolabs-erlangen.de/resources/MIR/ FMP/C6/C6S2 Tempō gramFourier.html，访问日期：2022-10-25。[14] Peter Grosche、Meinard M¨uller 和 Frank Kurth，《循环速度图——一种面向音乐信号的中层速度表示》，载于*2010* 年 *IEEE* 声学、语音与信号处理国际会议。IEEE，2010 年，第 5522–5525 页。[15] Frank Kurth、Thorsten Gehrmann 和 Meinard M¨uller，《循环节拍谱：用于时间尺度不变音频识别的速度相关音频特征》，载于国际音乐信息检索学会会议（*ISMIR*），2006 年，第 35–40 页。[16] Irene Gratton、Maria A Brandimonte 和 Nicola Bruno，《音乐家与非音乐家对速度的绝对记忆》，*PLOS ONE*，第 11 卷，第 10 期，第 e0163558 页，2016 年。[17] G. Tzanetakis 和 P. Cook，《音频信号的音乐流派分类》，*IEEE*《语音与音频处理汇刊》，第 10 卷，第 5 期，第 293–302 页，2002 年。[18]Ugo Marchand、Quentin Fresnel 和 Geoffroy Peeters，《GTZAN-Rhythm：为 GTZAN 测试集扩充节拍、强拍和摇摆节奏标注》，2015 年 10 月，发表于第 16 届国际音乐信息检索学会会议的最新成果演示环节，2015 年。[19] Franc

ˋois Chollet 等人，“Keras”，https://keras.io，2015 年。

[20] Diederik Kingma 和 Jimmy Ba，“Adam：一种随机优化方法”，国际学习表征会议 *(ICLR)*，2015 年。[21] Brian McFee、Colin Raffel、Dawen Liang、Daniel Ellis、Matt McVicar、Eric Battenberg 和 Oriol Nieto，“librosa：用于 Python 的音频与音乐信号分析”，2015 年，第 18–24 页。[22] Meinard M¨uller 和 Frank Zalkow，“libfmp：一款用于音乐处理基础的 Python 软件包”，《开放源代码软件杂志》，第 6 卷，第 63 期，第 3326 页，2021 年 7 月。[23] Pauli Virtanen 等人，“SciPy 1.0：Python 科学计算的基础算法”，《自然·方法》，第 17 卷，第 261–272 页，2020 年。