arXiv:2602.09891v1 [cs.SD] 10 Feb 2026

STEMPHONIC：灵活的多音轨同时音乐生成

Shih-Lun Wu[♯♭*] *Ge Zhu*[♭] *Juan-Pablo Caceres*[♭] *Cheng-Zhi Anna Huang*[♯] *Nicholas J. Bryan*[♭]

[♯] MIT CSAIL [♭] Adobe Research

摘要

音乐分轨生成旨在创作彼此同步且相互独立的乐器音频片段。与传统的文本生成音乐模型相比，这种方法有望为用户提供更高的控制力，并更好地契合音乐人的工作流程。然而，现有的分轨生成方法要么依赖固定架构，并行输出预定义的一组分轨；要么一次只生成一个分轨，尽管能够灵活组合分轨，却导致顺序推理速度缓慢。我们提出了 STEMPHONIC，这是一种基于扩散/流的框架，打破了这一取舍，只需一次推理过程即可生成数量可变的同步音轨。训练时，我们将每个音轨视为一个批次元素，在批次中对同步音轨进行分组，并为每组应用共享噪声潜变量。推理时，我们使用共享初始噪声潜变量和特定音轨的文本输入，一次性生成同步的多音轨输出。我们还进一步扩展该方法，以支持一次性条件式多声部生成和逐音轨活动控制，使用户能够迭代生成并编排混音的时间分层。我们在多个开源音轨评测集上对结果进行基准测试，结果表明，STEMPHONIC 能够生成质量更高的输出，同时将完整混音的生成过程加速 25–50%。演示见：
https://stemphonic-demo.vercel.app。

索引词—音乐音频生成、分轨生成、条件分轨生成、可变音轨组合、扩散、流。

1. 引言

文本到音频音乐生成模型如今能够根据简单的文本输入生成听起来逼真的音乐 [1–5]。它们降低了音乐创作的门槛，让任何人都能探索并表达自己的创造力，但通常生成的是完整混音的多乐器输出，难以编辑，也不便于作为组件在新乐曲中重复使用 [6, 7]。为了让创作者不再局限于文本提示，人们提出了许多控制与编辑方法，包括细粒度的时序控制 [8–10]，音乐修补 [10–12]，以及音乐分轨生成 [13–16]。音乐音轨是对一种或多种乐器的录音，它们共同构成一个独立的

混音中的音轨，例如，鼓 用于奠定节奏基础，贝斯 用于低音进行。生成分轨可让创作者分别编辑每条音轨，并尝试不同的混音和母带处理技术，从而增强创作掌控力 [17, 18]。

现有的分轨生成方法大致可分为（*1*）采用并行化架构的模型，以及（*2*）需要按顺序生成分轨的单分轨模型。并行化模型 [13–16] 可在单次处理中生成多个相互协调的分轨，但只能处理有限且粒度较粗的分轨类型（例如仅包含贝斯、鼓、人声和其他），这些类型必须预先确定并构建到架构中。另一方面，单分轨模型 [19–23]，则允许

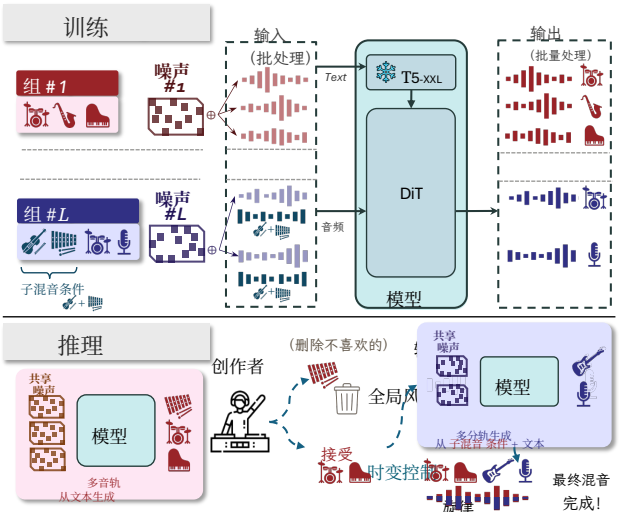

图 1。我们的 STEMPHONIC 框架可灵活生成多音轨音乐。（上）训练时，每组音乐同步分轨共享同一个噪声潜变量。（下）推理时，我们使用共享初始噪声，一次推理过程即可生成数量可变的多音轨输出。我们还支持条件式音轨生成和逐音轨活动控制。

通过文本提示或其他条件化机制灵活地生成开放词汇分轨，并且通常可以基于现有音频进行条件化，迭代生成新的伴奏分轨，从而创建包含任意数量分轨的混音。然而，这些模型一次生成一个分轨，导致完整的推理过程速度较慢。

为融合这两种范式各自的优势并弥补其不足，我们提出了 STEMPHONIC，一种基于潜在扩散/流的 [24,25]框架，能够在一次推理过程中生成数量可变的音乐同步分轨，如图 1所示。我们引入了两种训练阶段采用的技术。首先，将每条音轨视为一个批次元素，并在一个批次中对音乐同步分轨进行分组（第 3.1 节）。然后，为每组分配一个共享噪声潜变量（第 3.2 节）。推理时，我们使用共享初始噪声和不同的特定音轨文本输入，在一次推理过程中生成数量可变的同步音轨输出。我们进一步将该方法拓展至条件式多音轨生成，并加入逐音轨活动控制（第 3.3 节），使创作者能够迭代生成并编排混音的时间分层。在实验中，我们对分轨分组和噪声共享这两项核心技术进行了消融研究，并验证了条件式生成和逐音轨活动控制能力。与现有的逐音轨迭代工作流相比，我们发现 STEMPHONIC 能够生成质量更高的多音轨混音，同时将完整推理过程加速 25–50%。

我们的贡献总结如下：

- 一种潜在扩散／基于流的框架，可在一次推理过程中高效生成数量可变的同步音轨，

* 在 Adobe Research 实习期间完成的工作。

既可从头生成，也可基于现有音频进行条件生成。

• 一种用于精确控制各个分轨输出活跃度的方法。

• 通过定量评估和定性演示示例，展示高质量输出以及灵活的作曲家式工作流。

总的来说，我们结合并行化模型的速度与单分轨模型开放词汇的自由度，提供了一种高效且高度可控的分轨生成框架。

2. 背景

我们基于潜在扩散 [26] 和整流流（RF） [24]生成模型构建，这些模型采用 Transformer 主干网络 [27],，为简便起见，称为扩散 Transformer（DiT）。我们以在通用音乐混音上预训练的基础模型权重初始化 DiT，然后使用以下 RF目标函数在独立分轨上对其进行微调：

$$\min_{\theta} \mathop{\mathbb{E}}_{\substack{(\boldsymbol{x}_k, \mathcal{C}_k) \sim \text{data} \\ \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}_{\mathrm{T}D \times \mathrm{T}D}) \\ t \sim \text{LogitNormal}_{\mu,\sigma}(0,1)}} \left\| \boldsymbol{v}_\theta\left(\boldsymbol{x}_k(t), t, \mathcal{C}_k\right) - \left(\boldsymbol{x}_k - \boldsymbol{\epsilon}\right) \right\|_2^2, \quad (1)$$

其中， $\boldsymbol{x}_k \in \mathbb{R}^{\mathrm{T}\times D}$ 是要建模的变分自编码器（VAE）音频潜变量，k 是分轨索引，T 是 VAE 潜变量帧数（即时间维度）， D 是逐帧维度，θ 是可训练 DiT 参数集合， $\mathcal{C}_k$ 是与分轨 $\boldsymbol{x}_k$ 相关联的任意条件，用于实现用户控制（例如分轨类型、现有分轨的音频混音、文本描述和速度）， $\boldsymbol{\epsilon} \in \mathbb{R}^{\mathrm{T}\times D}$是高斯初始噪声， $t \in (0,1)$ 是采样得到的噪声级别（即扩散/流时间步）， $\boldsymbol{x}_k(t) := (1-t)\boldsymbol{x}_k + t\boldsymbol{\epsilon}$ 是模型接收的加噪 VAE 潜变量，而 $\boldsymbol{v}_\theta(\cdot)$ 是模型预测的速度，用于对潜变量进行降噪。推理过程中，我们求解相关的概率流常微分方程， $d\boldsymbol{x}_k(t) = \boldsymbol{v}_\theta\left(\boldsymbol{x}_k(t)\ \mathcal{C}_k, t\right)dt$，通过欧拉离散化 [28]，将初始高斯噪声确定性地转换为逼真数据。

3. 方法

我们提出一种新颖框架，能够一次性生成数量/集合可变的音乐同步分轨。我们通过两种主要技术干预训练批次构建来实现这一点，即分轨分组和噪声共享，从而将分轨间连贯性与同步的归纳偏置直接注入模型。

3.1. 分轨数据与分组

我们在由孤立分轨音频片段构成的数据集上微调 DiT。来自同一乐曲的分轨可以线性组合，形成连贯的音乐混音 $\boldsymbol{w}_{\text{mix}} := \sum_k (g_k \cdot \boldsymbol{w}_k)$，其中 $g_k \in \mathbb{R}$ 是每个单独音轨可由用户调节的增益，而 $\boldsymbol{w}_{\text{mix}}, \boldsymbol{w}_k$ 分别是混音与各分轨的原始音频波形。我们进一步将子混音定义为完整混音中一个或多个分轨子集的混合。我们使用相同的 VAE 音频到潜变量编码器以及潜变量到音频解码器 [3, 14, 15, 29],处理分轨、混音和子混音各自的立体声 44.1kHz 波形，从而获得用于分轨微调的 VAE 潜变量，即 $\boldsymbol{x}_k$。我们开发了一种分轨分组 机制，确保音乐同步分轨出现在同一个训练批次中。这与标准的批次构建机制形成对比，在后者中各个分轨被独立采样 [20–22]。

我们将 $\mathcal{M}$ 定义为由多音轨合成曲（或混音）组成的训练数据集，其中每个混音都由其组成音轨的 VAE 潜变量 $\mathcal{X}^{(m)} = \{\boldsymbol{x}_k^{(m)}\}_{k=1}^{K^{(m)}}$ 表示；$m \in \{1, \ldots, |\mathcal{M}|\}$ 用于索引混音， $K^{(m)}$ 是该 m^{th} 混音的组成音轨数量。我们还将 $\mathcal{B}$ 定义为待采样的训练批次（由音轨组成），其批次大小固定为 B。为构建 $\mathcal{B}$，我们运行一个循环，在每次迭代中，以 $l \in \mathbb{N}$为索引，采样一个混音 $m\ (l) \sim \{1, \ldots, |\mathcal{M}|\}$，然后从该混音的组成音轨中采样一个子集 $\tilde{\mathcal{X}}^{(l)} \subseteq \mathcal{X}^{(m(l))}$，以获得 $|\tilde{\mathcal{X}}^{(l)}|$ 个目标分轨作为批次元素。循环持续运行，直到目标分轨数量达到批次大小 B；因此，每个批次中的组数（记为 L）取决于所采样的组。我们从混音中采样子集，而不是始终包含完整混音，以组合方式增加数据多样性。

为了支持迭代创作工作流，我们还训练模型，使其能够以现有音轨或子混音为条件生成新的音轨，类似于 [21]。具体而言，我们会随机选取一个批次中一半的 L 组来执行此条件生成任务。对于选中的组，我们从对应混音中“留出”的音轨里采样一个子集，即 $\mathcal{X}^{(m(l))} \setminus \tilde{\mathcal{X}}^{(l)}$，将其波形相加，然后获取该子混音的 VAE 潜变量作为条件，即 $\boldsymbol{x}^{(l)}{}_{\text{cond}} \in \mathbb{R}^{\mathrm{T}\times D}$。请注意，这实际上是将子混音视为各输出音轨共享的条件的一部分 $\mathcal{C}_k$（参见公式（1）），对应组中的每个输出音轨均共享该条件 $\boldsymbol{x}_k^{(l)} \in \tilde{\mathcal{X}}^{(l)}$。我们将此条件与加噪后的目标音轨潜变量 $\boldsymbol{x}_k^{(l)}$ (t) 按通道维度拼接，作为 DiT 模型的输入。对于未选中的组（即用于从头生成任务的组），我们将 $\boldsymbol{x}^{(l)}{}_{\text{cond}}$ 填充为零。

3.2. 扩散/流噪声共享

即使采用了分轨分组，仍然存在一个挑战：同一批次中包含多个分轨组，每组对应一个不同的混音。由于每条独立音轨都是一个批次元素，在模型的计算图中，无论如何分组，每条音轨都是独立的。同组音轨之间唯一共享的信息来自 $\mathcal{C}_k$ 的部分内容，其中包括部分文本提示（详情见第 3.4 节），以及条件生成情况下的条件化子混音 $\boldsymbol{x}^{(l)}{}_{\text{条件化}}$。因此，需要额外的信号向模型更明确地提供分组信息，以便模型更好地学习分轨间对齐与编曲。

为此，我们对噪声采样过程进行干预。初始噪声 $\boldsymbol{\epsilon}$ 是一种高维变量，在以 $\mathcal{C}_k$ 为条件进行条件化后，为生成提供多样性的来源。按照惯例，假设批次元素相互独立同分布，即 *i.i.d.*，并据此独立采样 $\boldsymbol{\epsilon}$。然而，由于现在我们的批次包含相关联的音轨组，我们可以利用噪声潜变量的高维性（$\mathbb{R}^{\mathrm{T}\times D}$）作为强有力的信号来指示分组。具体而言，对于每个组 $l \in \{1, \ldots, L\}$，我们采样一个共享初始噪声 $\boldsymbol{\epsilon}^{(l)}$，并将其应用于组内所有音轨，即噪声与音轨的配对方式如下：

$$(\boldsymbol{\epsilon}^{(l)}, \boldsymbol{x}_k^{(l)})\ \forall \boldsymbol{x}_k^{(l)} \in \tilde{\mathcal{X}}^{(l)}; \quad \boldsymbol{\epsilon}^{(l)} \sim \mathcal{N}(0, \mathbf{I}_{\mathrm{T}D\times\mathrm{T}D}). \quad (2)$$

通过这种方式，我们确保同一组中的音乐同步分轨共享噪声，而不同组（满足*i.i.d.*）仍接收独立噪声，从而有力地强化分组属性。在推理时，我们为所有批次元素分配一个共享的随机初始噪声，以生成音乐同步分轨。虽然这里采用的是 RF目标函数，但我们的噪声共享技术通常适用于任何使用高维噪声潜变量的扩散或基于流的模型。

3.3. 逐音轨活动控制

除了生成同步良好的分轨外，为了赋予用户更多编曲自主权，尤其是在生成大量分轨时（例如 > 5），我们设计了一种逐音轨活动控制机制。用户可以指定每条音轨的时间活动，从而精确地叠加构建最终混音。

具体而言，我们对每条音轨的原始波形进行简单的基于响度的静音检测，$\boldsymbol{w}_k$。根据预设的响度阈值（例如 -60 dB），我们得到一个静音片段列表。随后，我们将其转换／量化为一个二值单通道序列，记为$\boldsymbol{a}_k \in \{0,1\}^{\mathrm{T}\times 1}$，其中 1 表示活跃，0 表示静音，并将其加入条件 $\mathcal{C}_k$。（注意，时间维度 T 与目标音轨潜变量的时间维度一致 $\boldsymbol{x}_k$。）我们为活动指示器1 和 0学习一个小型（16 维）嵌入，并将嵌入序列沿通道维度与加噪后的目标 $\boldsymbol{x}_k(t)$ 拼接，然后输入 DiT，这与子混音条件机制相对应（参见第 3.1节）。

3.4. 文本条件控制

我们的模型支持自由文本控制，与传统的文本生成音乐模型类似 [2,3,5]。与以往针对单个音轨进行文本条件生成 [19,21,22],或针对固定组合进行并行分轨生成的研究相比 [14,16],，我们的方法需要结合 全局 （即组内所有音轨共享）和 逐音轨 描述，并将它们写入一条文本指令中。我们将其形式化为：先写逐音轨部分，再写全局部分。提示词示例：“生成音轨： *[吉他]*（给定上下文音轨：*[鼓、贝斯]*），生成以下音乐描述所对应的音乐：轻松的乡村摇滚……”。“给定上下文音轨”子句仅用于条件生成（即非从头生成的情况），并且在训练时会随机丢弃。

4. 实验

4.1. 实现细节

我们的模型架构与 Stable Audio Open 类似 [30]——一个变分自编码器 [29] 将立体声 44.1 kHz 波形压缩为潜变量（具有 $D = 64$ 维度），帧率为 12 Hz。一个拥有十亿参数的扩散 Transformer（DiT） [27] 对变分自编码器潜变量进行建模，文本条件则通过交叉注意力以 T5-XXL [31]嵌入的形式输入 DiT。除第 3节介绍的条件外，模型还通过 bpm 值进行速度条件控制。所有条件均以 1/3 的概率独立丢弃，以支持无分类器引导（CFG） [32]。我们先使用完整音频混音进行预训练，再运用所提出的技术针对分轨进行微调。模型以 32 秒的片段进行训练（对应的帧数为 $\mathrm{T} = 394$），每个 GPU 的批次大小为 1024 秒，有效批次大小（每个梯度步）为 16K 秒。我们使用 AdamW 优化器 [33]，并采用恒定的 10^{-4} 学习率。分轨生成训练在 30K 个梯度步后收敛，在 8× A100（80G）GPU 上耗时 3 天。推理时，我们使用包含 32 步的一阶欧拉采样器。仅在步骤 3∼28 [34] 中对所有条件应用无分类器引导，CFG 缩放系数设为 3.0。

4.2. 数据集

我们使用 20K 小时的授权音乐混音进行预训练，然后使用对应于 ∼400 小时混音的授权分轨进行微调（每个混音含 ≈6 个分轨）。两个数据集都包含混音级别的文本描述和速度（bpm）元数据，音轨时长平均为 2.5 分钟。我们的分轨数据集包含 50 多种分轨类型，其中以下 11 种占绝大多数：鼓、贝斯、打击乐、合成器、键盘、吉他、弦乐、音效、人声、合成人声和管乐。评估时，按照近期文献 [22], 的做法，我们使用开源分轨数据集：MoiseDB [35] 和 MusDB [36],，两者都包含约 10 小时的混音。为构建评估样例，我们将混音裁剪为 32 秒的片段，并纳入片段中活跃度达到 ≥50% 的所有分轨。我们利用 Qwen2.5-Omni [37], 这一具备音乐分析能力的多模态大语言模型来获取文本描述，并使用 Madmom 节拍跟踪器 [38] 的内部版本估算速度（bpm）。我们主要通过规则将 MoiseDB 和 MusDB 中的分轨类型映射到上述排名前 11 的分轨类型；对于原本标记为“other*”的项目，则使用 Qwen2.5-Omni 进行标注，再通过人工检查和修正。MoisesDB 数据集中出现次数最多的分轨类型（>100 次）包括：鼓、贝斯、吉他、人声、钢琴、键盘、打击乐和弦乐。MusDB 中最常见的分轨类型为鼓、贝斯、人声、吉他、键盘和合成器。分段后，我们从 MoisesDB 中得到约 1.5K 个混音（即分组后的分轨），从 MusDB 中得到约 1K 个。MoisesDB 每个混音的平均分轨数为 4.6（标准差 1.3），MusDB 为 3.5（标准差 0.7）。

4.3. 基线与实验设置

我们的实验分为三组。在主要消融实验中，我们研究了在一次推理过程中生成数量可变且在音乐上同步的音轨这一核心能力。为此，我们采用音轨分组（第3.1节）和噪声共享（第3.2节）（将此设置称为**‘C’**），或仅采用音轨分组（**‘B’**），或两者均不采用（**‘A’**），由此训练出三个模型。请注意，无论训练时是否采用噪声共享，我们都可以在推理时让预期同步的音轨共享噪声，以此作为一种无需训练、促进分轨间连贯性的手段。因此，推理时有两种变体：共享噪声（称为**‘(ii)’**）或不共享（**‘(i)’**）。总而言之，我们的完整设置为**C-(ii)**，消融程度最高的设置**A-(i)**，则代表将现有单分轨模型简单扩展为多分轨生成的情形 [19,21,22]（仅做最少的操作性修改）。

接下来，我们注意到，使用基础的 **A-(i)** 设置，仍可通过执行 K 次推理，生成可变数量的分轨混音。K 因此，我们需要展示我们的 C-(ii) **设置带来的整体**工作流改进。因此，我们比较以下方式从头生成 K 音轨：（*1*） K **次推理，使用**A-(i)**；**（2）1次推理，使用**C-(ii)**；**以及（**3**）*2*次推理，使用C-(ii)。**后者是折中方案，既展现了模型从头生成和条件生成的能力，**[1] 也体现了其独特的灵活性，用户可根据需要选择任意数量的推理轮次。我们在此评估生成质量和总推理时间。

作为最后一项实验，我们训练了完整的**C** 模型，并加入逐音轨活跃度控制（第3.3节）。随后，我们评估两个关键方面：控制本身的有效性，以及在推理时启用或不启用控制的情况下，它对多分轨生成质量的影响。需要指出的是，所有推理过程中均未使用评估集中的真实分轨音频；我们只使用派生元数据（即 {text prompt, stem types, tempo}）作为输入。所有推理均在 1× A100 (80G) GPU 上完成。生成的音轨按照模型确定的相对响度（音轨之间）进行混音，然后统一归一化至 –16 dBFS。

4.4. 评估指标

我们从以下几个关键方面评估生成的音频。

• **分轨控制**：我们使用 VGGish 特征提取器计算参考音频与生成的 [39] 分轨音频之间的弗雷歇音频距离（FAD）。我们将 MoiseDB 或 MusDB 中每种分轨类型的所有音频分别视为一个参考集，并报告所有分轨类型的宏平均值。我们将此指标简称为 **FAD**音轨。

• **混音质量**：我们还测量 FAD（简称**FAD**混音），但在这里，我们将真实混音作为参考。请注意，这也

[1]第一轮生成一半音轨，第二轮则根据第一轮输出音轨的子混音生成剩余的一半。如果音轨数 K 为奇数，则第一轮多分配一条音轨。

表 1。关于**核心技术**的消融实验：音轨分组 和噪声共享。我们一**次性**生成混音中的所有组成音轨。完整设置**C-(ii)**总体上优于所有其他消融设置，在难度更高的 MoisesDB 数据集上增益尤其显著。（n是每个评估数据集中的混音数量。）

	训练		推理	MoisesDB (n = 1488) / MusDB (n = 964)		
				分轨控制	混音质量	混合文本控制
设置	*stem grp.*	噪声分享	噪声分享	**FAD**$_{\text{stem}}$ ↓	**FAD**$_{\text{mix}}$ ↓	**CLAP** ↑
A-(i)	✗	✗	✗	2.69 / 2.91	1.84 / 1.09	28.82 / 28.73
A-(ii)	✗	✗	✓	2.80 / 3.02	1.78 / 1.24	28.67 / 28.28
B-(i)	✓	✗	✗	2.41 / 2.92	1.55 / **0.91**	28.85 / 29.14
B-(ii)	✓	✗	✓	2.41 / 2.97	1.53 / 1.10	28.93 / 28.76
C-(ii)	✓	✓	✓	**2.31** / **2.72**	**1.25** / 1.05	**30.19** / **29.27**

表2。**生成 音轨混音时相较于传统迭代式**（**A-(i)**， K 次遍历）基线的工作流改进。我们的模型 **C-(ii)** 提供了独特的灵活性，可在速度与质量之间权衡：单次遍历设置可实现 50%+ 的加速，而两次遍历设置可获得整体最佳生成结果（**FAD**$_{\text{mix}}$ 和**CLAP**），同时仍可节省 25–50 % 的总推理时间。音轨组和提示词来自 MoisesDB。

		$K=3$ ($n=190$) / $K=4$ ($n=456$) $K=5$ ($n=379$) / $K=6$ ($n=283$)		
设置	# 推理遍	**FAD**$_{\text{mix}}$ ↓	**CLAP** ↑	**时间** （秒） ↓
A-(i)	K	1.03 / 1.02 1.48 / 2.09	**30.65** / 30.34 29.65 / **30.67**	4.10 / 5.50 6.88 / 8.28
C-(ii)	2	**0.78** / **0.83** **1.34** / **1.92**	30.62 / **31.23** **30.32** / 30.05	3.03 / 3.27 3.70 / 4.16
	1	1.06 / 1.12 1.56 / 2.29	29.51 / 30.44 30.26 / 29.93	**2.00** / **2.52** **3.13** / **3.60**

评估音轨兼容性，因为如果各分轨质量很高却不同步，混音听起来就会与任何真实混音都不一样。[2]

- **混音文本控制**：我们计算所生成混音的音频嵌入与全局文本描述的文本嵌入之间的成对余弦相似度，二者均来自对比训练的 **CLAP** 模型 [42], , 使用音乐检查点。对于逐音轨活动控制，我们在输出音频上运行静音检测算法以获得活动序列，并计算**逐帧 F1** 相对于输入控制的分数 $\boldsymbol{a}_k$。这种基于循环一致性的评估也被用于现有的时序控制 [8]。

5. 结果与讨论

核心单次生成消融实验的结果如表 1所示。我们完整的 **C-(ii)** 设置总体上产生了最佳生成结果，尤其是在更具挑战性的 MoisesDB 评估集上，其中每个混音包含更多分轨。从定性角度看，基线 **A-(i)**设置始终无法生成同步音轨，从而导致混音不连贯。中间设置（**A-(ii), B-(i), B-(ii)**）有显著改进，偶尔能生成音乐同步分轨，但其一致性远不如我们完整的 **C-(ii)** 设置。这在一定程度上表明（*1*）初始噪声确实捕捉了语义

[2]我们还研究了用于音轨兼容性的 COCOLA 指标 [41]，但发现该指标效果不佳，可能是因为其训练数据（主要为合成音频）与我们的数据（主要为录音室录制音频）之间存在领域不匹配。

表3。评估**音轨活跃度控制**。我们提出的机制（第3.3节）可实现近乎完美的活跃度控制（逐帧**F1**），并且在推理时可灵活选择是否启用，代价是其他指标仅受到适度影响。（**F1**仅针对部分时间处于活跃状态的音轨进行评估。）

	训练	推理。	MoisesDB (n = 1488) / MusDB (n = 964)			
	act ctrl	*act ctrl*	分轨控制	混音质量	混合文本控制	动作控制
设置			**FAD**$_{\text{stem}}$↓	**FAD**$_{\text{mix}}$↓	**CLAP**↑	**F1** (%)↑
C-(ii)	✗	✗	**2.31** / **2.72**	**1.25** / **1.05**	**30.19** / **29.27**	n.a.
C-(ii)	✓	✗	2.66 / 2.74	1.54 / 1.08	28.78 / 28.94	n.a.
+Act	✓	✓	2.47 / 2.77	1.46 / 1.13	29.55 / 29.14	**99.42** / **99.43**

在节奏与和声方面，因此仅在推理时共享噪声（即***-(ii)**）会强制一定程度的连贯性，并且（2）设置**B-(*)**中分组分轨所共享的信息（例如文本提示）也会促成分轨间的连贯性，尽管其效果不如在**C-(ii)**中共享高维噪声。

表2对比了 K 音轨工作流，分别采用**A-(i)** （类似于单分轨模型 [19,21,22]），以及我们的 **C-(ii)** 方案所支持的工作流。主要发现如下：给定 K 个推理轮次，**A-(i)**生成的混音质量可与我们的**"C-(ii)，1 轮"**工作流相当，甚至略胜一筹，但耗时至少是其 2× 倍。与此同时，我们的**"C-(ii)，2 轮"**工作流通常能生成质量最佳的结果，且 **FAD**$_{\text{mix}}$ 始终较低，同时比 **A-(i)** 快 25–50%，且时间优势随音轨数量增加而扩大 K。我们认为，2 轮工作流输出质量更优，源于训练过程中形成的归纳偏置——由于我们的子集采样策略（参见第 3.1 节）产生的组大小服从二项分布，且峰值位于 $K/2$，因此模型生成这一大小的组时表现天然更佳。

最后，表 3 展示了使用分轨活动控制训练的模型性能（参见第 3.3节）。帧级F1分数表明，我们的活动控制近乎完美。其他指标略有下降；从定性角度看，这主要是由于音频洁净度降低，而非分轨间连贯性受损，这是为增加控制选项并允许在推理时选择性应用这些控制所作出的权衡。我们欢迎读者访问演示网站 https://stemphonic-demo.vercel.app，其中展示了静态生成结果以及类似作曲家的生成结果；在后者中，作者以现实用户的身份迭代调整最终混音。

6. 结论与未来工作

我们提出了 STEMPHONIC，这是一种基于扩散/流的框架，能够单次生成可变组合的、在音乐上同步的音轨。为此，我们采用了所提出的训练阶段干预措施：分轨分组和噪声共享。我们还支持条件式多声部生成和逐音轨活动控制。实验表明，与现有迭代基线相比，STEMPHONIC 不仅将多分轨生成工作流加速了 25–50%，还生成了质量更高的混音。

我们的工作为未来研究开辟了若干前景广阔的方向。深入分析噪声共享机制的理论，或可带来根本性见解，帮助理解如何在生成模型中将高维噪声用作显式条件信号。在用户交互方面，超越简单的音轨类型标签，改用自由文本描述来描述各个音轨，仍是实现更精准控制的重要下一步。此外，研究如何控制分音轨音乐新颖性 [43], , 或如何根据全局文本提示由智能体提出音轨组合预设 [44]，都能显著提升用户参与度及此类系统的协作潜力。

7. 参考文献

[1] Andrea Agostinelli、Timo I Denk、Zalán Borsos、Jesse Engel 等人，《MusicLM：从文本生成音乐》，arXiv:2301.11325，2023。[2] Jade Copet、Felix Kreuk、Itai Gat、Tal Remez、David Kant、Gabriel Synnaeve、Yossi Adi 和 Alexandre Défossez，《简单且可控的音乐生成》，发表于 NeurIPS，2023。[3] Zach Evans、CJ Carr、Josiah Taylor、Scott H Hawley 和 Jordi Pons，《快速时间条件潜在音频扩散》，发表于 ICML，2024。[4] Zachary Novack、Ge Zhu、Jonah Casebeer、Julian McAuley、Taylor Berg- Kirkpatrick 和 Nicholas J. Bryan，《Presto!：蒸馏步骤与层以加速音乐生成》，发表于 ICLR，2025。[5] Antoine Caillon 、 Brian McWilliams 、 Cassie Tarakajian、Ian Simon、Ilaria Manco、Jesse Engel、Noah Constant、Pen Li、Timo I Denk 等人，《实时音乐模型》，arXiv:2508.04651，2025。[6] Chris Donahue、Shih-Lun Wu、Yewon Kim、Dave Carlton、Ryan Miyakawa 和 John Thickstun，《Hookpad Aria：歌曲创作者的智能助手》，发表于 ISMIR 最新成果演示，2024。[7] Yewon Kim、Sung-Ju Lee 和 Chris Donahue，《Amuse：借助多模态灵感实现人机协作歌曲创作》，发表于 CHI，2025。[8] Shih-Lun Wu、Chris Donahue、Shinji Watanabe 和 Nicholas J. Bryan，《Music ControlNet：用于音乐生成的多种时变控制》，发表于 IEEE/ACM T-ASLP，2024。[9] Or Tal、Alon Ziv、Itai Gat、Felix Kreuk 和 Yossi Adi，《用于时间可控文本生成音乐的音频与符号联合条件化》，发表于 ISMIR，2024。[10] Fang-Duo Tsai、Shih-Lun Wu、Weijaw Lee、Sheng-Ping Yang、Bo-Rui Chen、Hao-Chung Cheng 和 Yi-Hsuan Yang，《MuseControlLite：借助轻量级条件器实现多功能音乐生成》，发表于 ICML，2025。[11] Zachary Novack、Julian McAuley、Taylor Berg-Kirkpatrick 和 Nicholas J. Bryan，《DITTO：用于音乐生成的扩散推理时 t 优化》，发表于 ICML，2024。[12] Fang-Duo Tsai、Shih-Lun Wu、Haven Kim、Bo-Yu Chen、Hao-Chung Cheng 和 Yi-Hsuan Yang，《音频提示适配器：通过轻量微调释放文本生成音乐的编辑能力》，发表于 ISMIR，2024。[13] Giorgio Mariani、Irene Tallini、Emilian Postolache、Michele Mancusi、Luca Cosmo 和 Emanuele Rodolà，《用于同时进行音乐生成与分离的多源扩散模型》，发表于 ICLR，2024。[14] Yao Yao、Peike Li、Boyu Chen 和 Alex Wang，《JEN-1 Composer：高保真多轨音乐生成的统一框架》，发表于 AAAI，2025。[15] Tornike Karchkhadze、Mohammad Rasool Izadi 和 Shlomo Dubnov，《使用多轨潜在扩散模型同步进行音乐分离与生成》，发表于 ICASSP，2025。[16] Simon Rouard、Robin San Roman、Yossi Adi 和 Axel Roebel，《MusicGen-Stem：通过自回归建模实现多音轨音乐生成与编辑》，发表于 ICASSP，2025。[17] Martin E Malandro，《Composer’s Assistant：用于多轨 MIDI 补全的交互式 Transformer》，发表于 ISMIR，2023。[18] Martin E Malandro，《Composer’s Assistant 2：具有细粒度用户控制的交互式多轨 MIDI 补全》，发表于 ISMIR，2024。[19] Bing Han、Junyu Dai、Weituo Hao、Xinyan He、Dong Guo、Jitong Chen、Yuxuan Wang、Yanmin Qian 和 Xuchen Song，《InstructME：基于指令引导、采用潜扩散模型的音乐编辑与混音框架》，发表于 IJCAI，2024。[20] Julian Parker、Janne Spijkervet、Katerina Kosta、Furkan Yesiler、Boris Kuznetsov、Ju-Chiang Wang、Matt Avent、Jitong Chen 和 Duc Le，《StemGen：能够聆听的音乐生成模型》，发表于 ICASSP，2024。[21] Javier Nistal、Marco Pasini、Cyran Aouameur、Maarten Grachten 和 Stefan Lattner，《Diff-A-Riff：通过潜在扩散模型协同创作音乐伴奏》，发表于 ISMIR，2024。[22] Yunkee Chae 和 Kyogu Lee，《MGE-LDM：用于同时进行音乐生成与声源提取的联合潜在扩散》，载于 NeurIPS，2025 年。[23] Giorgio Strano、Chiara Ballanti、Donato Crisostomi、Michele Mancusi、Luca Cosmo 和 Emanuele Rodolà，《Stage：通过基于前缀的条件控制生成分轨伴奏》，载于 ISMIR，2025 年。[24] Xingchao Liu、Chengyue Gong 等，《让流笔直且快速：利用整流流学习生成和迁移数据》，载于 ICLR，2023 年。[25] Patrick Esser、Sumith Kulal、Andreas Blattmann、Rahim Entezari、Jonas Müller 等，《扩展用于高分辨率图像合成的整流流变换器》，载于 ICML，2024 年。[26] Robin Rombach、Andreas Blattmann、Dominik Lorenz、Patrick Esser 和 Björn Ommer，《利用潜在扩散模型进行高分辨率图像合成》，载于 CVPR，2022 年。[27] William Peebles 和 Saining Xie，《采用变换器的可扩展扩散模型》，载于 CVPR，2023 年。[28] Jiaming Song、Chenlin Meng 和 Stefano Ermon，《去噪扩散隐式模型》，arXiv:2010.02502，2020 年。[29] Jonah Casebeer、Ge Zhu、Zhepei Wang 和 Nicholas J. Bryan，《生成优先的神经音频自编码器》，载于 ICASSP，2026 年。[30] Zach Evans、Julian D Parker、CJ Carr、Zack Zukowski、Josiah Taylor 和 Jordi Pons，《Stable Audio Open》，载于 ICASSP，2025 年。[31] Colin Raffel、Noam Shazeer、Adam Roberts、Katherine Lee、Sharan Narang、Michael Matena、Yanqi Zhou、Wei Li 和 Peter J Liu，《探索迁移学习的极限：统一的文本到文本变换器》，《机器学习研究杂志》（JMLR），2020 年。[32] Jonathan Ho 和 Tim Salimans，《无分类器扩散引导》，arXiv:2207.12598，2022 年。[33] Ilya Loshchilov 和 Frank Hutter，《解耦权重衰减正则化》，arXiv:1711.05101，2017 年。[34] Tuomas Kynkäänniemi、Miika Aittala、Tero Karras、Samuli Laine、Timo Aila 和 Jaakko Lehtinen，《在有限区间内应用引导可改善扩散模型中的样本质量与分布质量》，载于 NeurIPS，2024 年。[35] Igor Pereira、Felipe Araújo、Filip Korzeniowski 和 Richard Vogl，《MoisesDB：一个超越 4 音轨的声源分离数据集》，载于 ISMIR，2023 年。[36] Zafar Rafii、Antoine Liutkus、Fabian-Robert Stöter、Stylianos Ioannis Mimilakis 和 Rachel Bittner，《用于音乐分离的 MUSDB18 语料库》，2017 年。[37] Jin Xu、Zhifang Guo、Jinzheng He、Hangrui Hu、Ting He、Shuai Bai、Keqin Chen、Jialin Wang、Yang Fan、Kai Dang 等，《Qwen2.5-omni 技术报告》，arXiv:2503.20215，2025 年。[38] Sebastian Böck、Filip Korzeniowski、Jan Schlüter、Florian Krebs 和 Gerhard Widmer，《Madmom：一种新的 Python 音频与音乐信号处理库》，载于 ACM MM，2016 年。[39] Kevin Kilgour、Mauricio Zuluaga、Dominik Roblek 和 Matthew Sharifi，《弗雷歇音频距离：一种用于评估音乐增强算法的指标》，载于 Interspeech，2019 年。[40] Shawn Hershey、Sourish Chaudhuri、Daniel PW Ellis、Jort F Gemmeke、Aren Jansen 等，《用于大规模音频分类的卷积神经网络架构》，载于 ICASSP，2017 年。[41] Ruben Ciranni、Giorgio Mariani、Michele Mancusi、Emilian Postolache、Giorgio Fabbro、Emanuele Rodolà 和 Luca Cosmo，《COCOLA：音乐音频表征的面向一致性的对比学习》，载于 ICASSP，2025 年。[42] Yusong Wu、Ke Chen、Tianyu Zhang、Yuchen Hui、Taylor Berg-Kirkpatrick 和 Shlomo Dubnov，《通过特征融合与从关键词到描述的增强进行大规模对比式语言—音频预训练》，载于 ICASSP，2023 年。[43] Mathias Rose Bjare、Stefan Lattner 和 Gerhard Widmer，《通过信息含量曲线匹配控制音乐生成中的意外度》，载于 ISMIR，2024 年。[44] Qixin Deng、Qikai Yang、Ruibin Yuan 等，《ComposerX：利用大语言模型进行多智能体符号音乐创作》，arXiv:2404.18081，2024 年。