Music ControlNet: Multiple Time-Varying Controls for Music Generation

Shih-Lun Wu, Chris Donahue, Shinji Watanabe, *Fellow, IEEE*, and Nicholas J. Bryan, *Member, IEEE*

***Abstract*—Text-to-music generation models are now capable of generating high-quality music audio in broad styles. However, text control is primarily suitable for the manipulation of *global* musical attributes like genre, mood, and tempo, and is less suitable for precise control over *time-varying* attributes such as the positions of beats in time or the changing dynamics of the music. We propose Music ControlNet, a diffusion-based music generation model that offers multiple precise, time-varying controls over generated audio. To imbue text-to-music models with time-varying control, we propose an approach analogous to pixel-wise control of the image-domain ControlNet method. Specifically, we extract controls from training audio yielding paired data, and fine-tune a diffusion-based conditional generative model over audio spectrograms given melody, dynamics, and rhythm controls. While the image-domain Uni-ControlNet method already allows generation with any subset of controls, we devise a new masking strategy to allow creators to input controls that are only partially specified in time. We evaluate both on controls extracted from audio and controls we expect creators to provide, demonstrating that we can generate realistic music that corresponds to control inputs in both settings. While few comparable music generation models exist, we benchmark against MusicGen, a recent model that accepts text and melody input, and show that our model generates music that is 49% more faithful to input melodies despite having 35x fewer parameters, training on 11x less data, and enabling two additional forms of time-varying control. Sound examples can be found at https://musiccontrolnet.github.io/web/.**

***Index Terms*—Music generation, controllable generative modeling, diffusion models.**

I. Introduction

ONE of the pillars of musical expression is the communication of high-level ideas and emotions through precise manipulation of lower-level attributes like notes, dynamics, and rhythms. Recently, there has been an explosion of interest in training text-to-music generative models that allow creators to directly convert high-level intent (expressed as text) into music audio [1], [2], [3], [4], [5]. These models suggest an exciting new paradigm of musical expression wherein creators can instantaneously generate realistic music without the need to write a melody, specify meter and rhythm, or orchestrate instruments. However, while dramatically more efficient, this new paradigm ignores more conventional forms of musical expression rooted in the manipulation of lower-level attributes, limiting the ability to express precise musical intent or leverage models in existing creative workflows. There are two primary obstacles for adding precise control to text-based music generation methods. Firstly, relative to symbolic music representations like scores, text is a cumbersome interface for conveying precise musical attributes that vary over time. Verbose and mundane text descriptions may be needed to precisely represent even the first note of a musical score e.g., "the song starts at 80 beats per minute with a quarter note on middle C played mezzo-forte on the saxophone". The second obstacle is an empirical one—text-to-music models tend to faithfully interpret *global* stylistic attributes (e.g., genre and mood) from text, but struggle to interpret text descriptions of precise musical attributes (e.g., notes or rhythms). This is perhaps a consequence of the relative scarcity of precise descriptions in the training data.

A potential solution to the lack of precision of natural language is the incorporation of *time-varying* controls into music generation. For example, one body of work looks at synthesizing music audio from time-varying symbolic music representations like MIDI [6], [7], however this approach offers a particularly strict form of control requiring users to compose entire pieces of music beforehand. Such approaches are more similar to typical music composition processes and do not take full advantage of recent text-to-music methods. Another body of work on musical style transfer [8], [9], [10], [11], [12], [13] seeks to transform recordings from one *style* (e.g., genre, musical ensemble, or mood) to another while preserving the underlying composition content. However, a majority of these approaches require training an individual model per style, as opposed to the flexibility of using text to control style in a single model.

In this work, we propose Music ControlNet, a diffusion-based music generation model that offers multiple time-varying controls over the melody, dynamics, and rhythm of generated audio, in addition to global text-based style control as shown in Fig. 1. To incorporate such time-varying controls, we adapt recent work on image generation with spatial control, namely, ControlNet [14] and Uni-ControlNet [15] to enable musical controls that are *composable* (i.e., can generate music corresponding to any subset of controls) and further allow creators to

Manuscript received 12 November 2023; revised 17 April 2024; accepted 21 April 2024. Date of current version 22 May 2024. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Juhan Nam. *(Corresponding authors: Shih-Lun Wu; Nicholas J. Bryan.)*

Shih-Lun Wu was an intern at Adobe Research, San Francisco, CA 94103 USA. He is now with the School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: shihlunw@cs.cmu.edu).

Chris Donahue is with the School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213 USA.

Shinji Watanabe is with the Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA 15213 USA.

Nicholas J. Bryan is with Adobe Research, San Francisco, CA 94103 USA (e-mail: njb@ieee.org).

Digital Object Identifier 10.1109/TASLP.2024.3399026

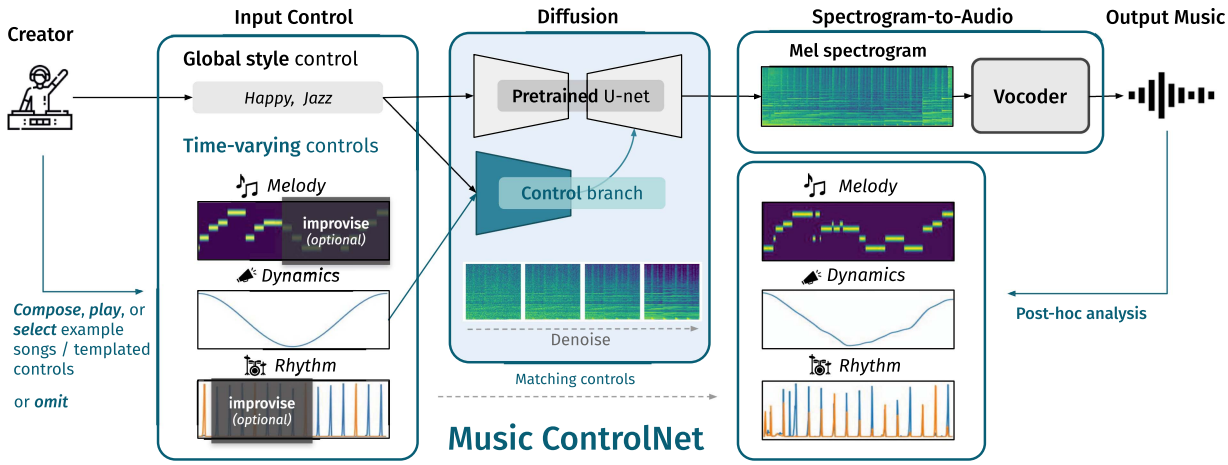

Fig. 1. Music ControlNet overview. Our model accepts as input global genre and mood text control, alongside any combinations of precise, time-varying melody, dynamics, and rhythm controls. The controls can each be fully or partially specified in time, the latter of which signals the model to musically improvise. Music that adheres to input controls is generated using a diffusion model that outputs an image-like representation of music (a Mel spectrogram), which is then rendered as audio using a vocoder. Music ControlNet empowers creators to blend text and musical controls in their creative process with a straightforward pipeline.

only *partially specify* each of the controls both for convenience and to direct our model to musically *improvise* in remaining time spans of the generation. To overcome the aforementioned scarcity of precise, ground-truth control inputs, following [5], [16], we extract useful control signals directly from music during training. We evaluate our method on two different categories of control signals: (1) *extracted* control signals that come from example songs, which are similar to those seen during training, and (2) *created* control signals that we anticipate musician creators might want to use in a co-creation setting via drawing or similar user-interface tools. Our experiments show that we can generate realistic music that accurately corresponds to control inputs in both settings. Moreover, we compare our approach against the melody control of the recently proposed MusicGen [5], showing that our model is 49% more faithful to melody input, despite also controlling dynamics and rhythm, having 35x fewer parameters, and being trained on 11x less data.

Our contributions include:

- A general framework for augmenting text-to-music models with composable, precise, time-varying musical controls.
- A method to enable one or more partially-specified time-varying controls at inference.
- Effective application of our framework to melody, dynamics, and rhythm control using music feature extraction algorithms together with conditional diffusion models.
- Demonstration that our model generalizes from extracted controls seen during training to ones we expect from creators.

II. Background: Diffusion and Image Generation

A. Diffusion Models

We use denoising diffusion probabilistic models (DDPMs) [17], [18] as our underlying generative modeling approach for music audio. DDPMs are a class of latent generative variable model. A DDPM generates data $\boldsymbol{x}^{(0)} \in \mathcal{X}$ from Gaussian noise $\boldsymbol{x}^{(M)} \in \mathcal{X}$ through a denoising Markov process that produces intermediate latents $\boldsymbol{x}^{(M-1)}, \boldsymbol{x}^{(M-2)}, \ldots, \boldsymbol{x}^{(1)} \in \mathcal{X}$, where $\mathcal{X}$ is the data space. DDPMs can be formulated as the task of modeling the joint probability distribution of the desired output data $\boldsymbol{x}^{(0)}$ and all intermediate latent variables, i.e.,

$$p_\theta\left(\boldsymbol{x}^{(0)}, \ldots, \boldsymbol{x}^{(M)}\right) := p\left(\boldsymbol{x}^{(M)}\right) \prod_{m=1}^{M} p_\theta\left(\boldsymbol{x}^{(m-1)} | \boldsymbol{x}^{(m)}\right), \tag{1}$$

where θ denotes learned parameters, and $p(\boldsymbol{x}^{(M)}) := \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ is a fixed noise prior, and m is the diffusion time index.

To create training examples, a forward diffusion process $q(\boldsymbol{x}^{(0)}, \ldots, \boldsymbol{x}^{(M)})$ is used to gradually corrupt clean data examples $\boldsymbol{x}^{(0)}$ via a Markov chain that iteratively adds noise:

$$\begin{aligned} q\left(\boldsymbol{x}^{(0)}, \ldots, \boldsymbol{x}^{(M)}\right) &:= q\left(\boldsymbol{x}^{(0)}\right) \prod_{m=1}^{M} q\left(\boldsymbol{x}^{(m)} | \boldsymbol{x}^{(m-1)}\right) \\ q\left(\boldsymbol{x}^{(m)} | \boldsymbol{x}^{(m-1)}\right) &:= \mathcal{N}\left(\sqrt{1-\beta_m}\boldsymbol{x}^{(m-1)}, \beta_m \boldsymbol{I}\right), \end{aligned} \tag{2}$$

where $q(\boldsymbol{x}^{(0)})$ is the true data distribution, and $\beta_1, \ldots, \beta_M$ are a sequence of parameters that define the noise level within the forward diffusion process, also known as the noise schedule.

By definition of $q(\boldsymbol{x}^{(m)} | \boldsymbol{x}^{(m-1)})$, it follows that the noised data $\boldsymbol{x}^{(m)}$ at any noise level $m \in \{1, \ldots, M\}$ can be sampled in one step via:

$$\boldsymbol{x}^{(m)} := \sqrt{\bar{\alpha}_m}\boldsymbol{x}^{(0)} + \sqrt{1-\bar{\alpha}_m}\boldsymbol{\epsilon}, \tag{3}$$

where $\bar{\alpha}_m := \prod_{m'=1}^{m}(1-\beta_{m'})$, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, and M is the total number of noise levels or steps during training. It was shown by Ho et al. [18] that we can optimize the variational lower bound [19] of the data likelihood, i.e., $p_\theta(\boldsymbol{x}^{(0)})$, by training a function approximator, e.g., a neural network, $f_\theta(\boldsymbol{x}^{(m)}, m) : \mathcal{X} \times \mathbb{N} \to \mathcal{X}$ to recover the noise $\boldsymbol{\epsilon}$ added via (3). More specifically, $f_\theta(\boldsymbol{x}^{(m)}, m)$ can be trained by minimizing the mean squared error, i.e.,

$$\mathbb{E}_{\boldsymbol{x}^{(0)}, \boldsymbol{\epsilon}, m}\left[\|\boldsymbol{\epsilon} - f_\theta(\boldsymbol{x}^{(m)}, m)\|_2^2\right]. \tag{4}$$

With a trained f_θ, we can transform random noise $\boldsymbol{x}^{(M)} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ to a realistic data point $\boldsymbol{x}^{(0)}$ through M denoising iterations. To obtain high-quality generations, a large M (e.g., 1000) is typically used. To reduce computational cost, denoising diffusion implicit models (DDIM) [20] further proposed an alternative formulation that allows running much fewer than M sampling steps (e.g., $50 \sim 100$) at inference with minimal impact on generation quality.

B. UNet Architecture for Image Diffusion Models

Our approach to music generation is rooted in methodology developed primarily for generative modeling of images. When applying diffusion modeling for image generation, the function f_θ is often a large UNet [18], [21]. The UNet architecture consists of two halves, an encoder and a decoder, that typically input and output image-like feature maps in the pixel space [22] or some learned latent space [23]. The encoder progressively downsamples the input to learn useful features at different resolution levels, while the decoder, which has a mirroring architecture to the encoder and accepts features from corresponding encoder layers through skip connections, progressively upsamples the features to eventually get back to the input dimension. For practical use, diffusion-based image generation models are often text-conditioned, which requires augmenting the network f_θ to accept a text description $\boldsymbol{c}_{\text{text}} \in \mathcal{T}$, where $\mathcal{T}$ is the set of all text descriptions. This leads to the following function signature:

$$f_\theta\left(\boldsymbol{x}^{(m)}, m, \boldsymbol{c}_{\text{text}}\right) : \mathcal{X} \times \mathbb{N} \times \mathcal{T} \to \mathcal{X}\,, \tag{5}$$

which, via the process outlined in Section II-A, models the desired probability distribution $p_\theta(\boldsymbol{x}^{(0)} \mid \boldsymbol{c}_{\text{text}})$. The text condition $\boldsymbol{c}_{\text{text}}$ is typically a sequence of embeddings from a large language model (LLM) or one or more embeddings from a learned embedding layer for class-conditional control. In either case, the conditioning signals m (i.e., the diffusion time step) and $\boldsymbol{c}_{\text{text}}$ are usually incorporated in the UNet hidden layers via additive sinusoidal embeddings [18] and/or cross-attention [23].

C. Classifier-Free Guidance

To improve the flexibility of text conditioning, classifier-free guidance (CFG) is commonly employed. CFG is used to simultaneously learn a conditional and unconditional generative model together and trade-off conditioning strength, mode coverage, and sample quality [24]. Practically speaking, during training CFG is achieved by randomly setting conditioning information to a special null value $\boldsymbol{c}_\emptyset$ for a fraction of the time during training. Then during inference, an image is generated using conditional control inputs, unconditional control inputs, or a linear combination of both. In most cases, a forward pass of $f_\theta(\boldsymbol{x}^{(m)}, m, \boldsymbol{c}_{\text{text}})$ and $f_\theta(\boldsymbol{x}^{(m)}, m, \boldsymbol{c}_\emptyset)$ per sampling step are needed and subsequent weighted averaging.

D. Adding Pixel-Level Controls to Image Diffusion Models

ControlNet [14] proposed an effective method to add *pixel-level* (i.e., *spatial*) controls to large-scale pretrained text-to-image diffusion models. Let the diffusion model input/output space be images, i.e., $\mathcal{X} := \mathbb{R}^{W \times H \times D}$, where W, H, D are respectively the width, height, and depth (for RGB images, $D = 3$) of an image, we denote the set of N pixel-level controls, indexed via $^{(n)}$, as:

$$\boldsymbol{C} := \left\{\boldsymbol{c}^{(n)} \in \mathbb{R}^{W \times H \times D_n}\right\}_{n=1}^{N}\,, \tag{6}$$

where D_n is the depth specific to each $\boldsymbol{c}^{(n)}$. For each condition signal $\boldsymbol{c}^{(n)}$, every pixel $\boldsymbol{c}^{(n)}_{i,j} \in \mathbb{R}^{D_n}$, where $i \in \{1, \ldots, W\}$ and $j \in \{1, \ldots, H\}$, asserts an attribute on the corresponding pixel $\boldsymbol{x}^{(0)}_{i,j}$ in the output image. For example, "$\boldsymbol{x}^{(0)}_{i,j}$ is (not) part of an edge" or "the perceptual depth of $\boldsymbol{x}^{(0)}_{i,j}$". Naturally, the function to be learned, f_θ, should be revised again as:

$$f_\theta\left(\boldsymbol{x}^{(m)}, m, \boldsymbol{c}_{\text{text}}, \boldsymbol{C}\right) : \mathcal{X} \times \mathbb{N} \times \mathcal{T} \times \mathcal{C} \to \mathcal{X}\,, \tag{7}$$

where $\mathcal{C}$ denotes the set of all possible sets of control signals. The updated f_θ hence implicity models $p_\theta(\boldsymbol{x}^{(0)} \mid \boldsymbol{c}_{\text{text}}, \boldsymbol{C})$.

To promote training data efficiency, ControlNet instantiates $f_\theta(\boldsymbol{x}^{(m)}, m, \boldsymbol{c}_{\text{text}}, \boldsymbol{C})$ by reusing the pretrained (and frozen) text-conditioned UNet, and clones its encoder half to form an adaptor branch to incorporate pixel-level control through finetuning. To gracefully bring in the information from pixel-level control, it enters the adaptor branch through a convolution layer that is initialized to zeros (i.e., a *zero convolution* layer). Outputs from layers of the adaptor branch are then fed back to the corresponding layers of the frozen pretrained decoder, also through zero convolution layers, to influence the final output. Uni-ControlNet [15] then augmented the adaptor branch such that one model can be finetuned to accept multiple pixel-level controls via a single adaptor branch without the need to specify all controls at once whereas ControlNet requires separate adaptor branches per control.

III. Music ControlNet

Our Music ControlNet framework builds on the methodology of text-to-image generation with pixel-level controls, i.e., ControlNet [14] and Uni-ControlNet [15], and extends it for text-to-audio generation with time-varying controls. We formulate our controllable audio generation task, explain the links and differences to ControlNet, and detail our essential model architecture and training modifications below.

A. Problem Formulation

Our overall goal is to learn a conditional generative model $p(\boldsymbol{w} \mid \boldsymbol{c}_{\text{text}}, \boldsymbol{C})$ over audio waveforms $\boldsymbol{w}$, given a global (i.e., time-independent) text control $\boldsymbol{c}_{\text{text}}$, and a set of time-varying controls $\boldsymbol{C}$. Due to our dataset, we limit $\boldsymbol{c}_{\text{text}}$ to *musical genre* and *moods* tags. Waveforms $\boldsymbol{w}$ are vectors in $\mathbb{R}^{\mathrm{T f_s}}$, where T is the length of audio in seconds and $\mathrm{f_s}$ is the sampling rate (i.e., number of samples per second). As $\mathrm{f_s}$ is large (typically between 16 kHz and 48 kHz), it is empirically difficult to directly model $p(\boldsymbol{w} \mid \cdot)$. Hence, we adopt a common hierarchical approach of using spectrograms as an intermediary. A *spectrogram* $\boldsymbol{s} \in \mathbb{R}^{\mathrm{T f_k} \times B \times D}$ is an image-like representation for audio signals, obtained through Fourier Transform on $\boldsymbol{w}$, where $\mathrm{f_k}$ is

the frame rate (usually 50∼100 per second), B is the number of frequency bins, and $D = 1$ for mono-channel audio. With $\boldsymbol{s}$ as the intermediary, we instead model the joint distribution $p(\boldsymbol{w}, \boldsymbol{s} \mid \boldsymbol{c}_{\text{text}}, \boldsymbol{C})$, which can be factorized as:

$$p(\boldsymbol{w}, \boldsymbol{s} \mid \boldsymbol{c}_{\text{text}}, \boldsymbol{C}) = p(\boldsymbol{w} \mid \boldsymbol{s}, \boldsymbol{c}_{\text{text}}, \boldsymbol{C}) \cdot p(\boldsymbol{s} \mid \boldsymbol{c}_{\text{text}}, \boldsymbol{C}) \quad (8)$$

$$:= p_\phi(\boldsymbol{w} \mid \boldsymbol{s}) \cdot p_\theta(\boldsymbol{s} \mid \boldsymbol{c}_{\text{text}}, \boldsymbol{C}), \quad (9)$$

where ϕ and θ are sets of parameters to be learned. Note that this factorization assumes conditional independence between waveform $\boldsymbol{w}$ and all control signals $\boldsymbol{c}_{\text{text}}$ and $\boldsymbol{C}$ given spectrogram $\boldsymbol{s}$, which is reasonable if the time-varying controls in $\boldsymbol{C}$ vary at a rate no faster than f_k by nature.

In our work, **we focus on modeling spectrograms given controls**, i.e., $p_\theta(\boldsymbol{s} \mid \boldsymbol{c}_{\text{text}}, \boldsymbol{C})$, and directly apply the DiffWave *vocoder* [25] to model $p_\phi(\boldsymbol{w} \mid \boldsymbol{s})$. Following the text-to-image ControlNet [14] model, we leverage diffusion models [18] to learn $p_\theta(\boldsymbol{s} \mid \boldsymbol{c}_{\text{text}}, \boldsymbol{C})$. If we set the input space $\mathcal{X} := \mathbb{R}^{\text{Tf}_\text{k} \times B \times D}$, and the desired output $\boldsymbol{x}^{(0)} := \boldsymbol{s}$, we can instantiate a neural network f_θ having an identical function signature to (7). However, we find two key differences between pixel controls for images and time-varying controls for music.

First, the first two dimensions in a spectrogram $\boldsymbol{s}$ have different semantic meanings, one being *time* and the other being *frequency*, as opposed to both being spatial in an image. Second, the time-varying controls useful to creators are closely coupled with *time*, but could have a much more relaxed relationship with *frequency* such that the second dimension of (6) cannot be restricted to B. For example, an intuitive control over 'musical dynamics' may involve defining volume over time, not over frequency. A high dynamics value for one frame can mean a number of different profiles over the B frequency bins for the corresponding spectrogram frame, e.g., a powerful bass playing a single pitch, or a rich harmony of multiple pitches, which the model has freedom to decide. Therefore, we relax the definition for the set of N control signals to become:

$$\boldsymbol{C} := \left\{ \boldsymbol{c}^{(n)} \in \mathbb{R}^{\text{Tf}_\text{k} \times B_n \times D_n} \right\}_{n=1}^{N}, \quad (10)$$

where B_n is the number of classes specific to each control $\boldsymbol{c}^{(n)}$, which is not bound to B. With this updated definition, the correspondence between control signals $\boldsymbol{C}$ and the output spectrogram $\boldsymbol{x}$ naturally becomes *frame-wise*. For example, suppose $\boldsymbol{c}^{(n)}$ represents dynamics control, a frame for the control $\boldsymbol{c}_t^{(n)} \in \mathbb{R}^{1 \times 1}$, where $t \in \{1, \ldots, \text{Tf}_\text{k}\}$, then describes "the musical dynamics (intensity) of the spectrogram frame $\boldsymbol{s}_t$".

Finally, we consider time-varying controls $\boldsymbol{c}^{(n)}$ that can be directly extracted from spectrograms. Given that spectrograms are also computed directly from waveforms, only pairs of $(\boldsymbol{w}, \boldsymbol{c}_{\text{text}})$ are necessary for training, causing no extra annotation overhead. Nevertheless, we note that our formulation supports manually annotated time-varying controls as well.

B. Adding Time-Varying Controls to Diffusion Models

We propose a strategy to learn the mapping between input controls, vectors of size B_n per time frame, to frequency bins, i.e., B in the output spectrograms, marking an update from ControlNet [14]. As mentioned in Section II-D, ControlNet clones the encoder half of the pretrained UNet for text-to-image generation as the *adaptor* branch, which uses newly attached zero convolution layers to enable pixel-level control. Let $\tilde{f}^{(l)}(\boldsymbol{x}^{(m,l-1)}, m, \boldsymbol{c}_{\text{text}}, \boldsymbol{C})$ be the l^{th} block of the adaptor branch where˜denotes the adaptor (not the main UNet), $\boldsymbol{x}^{(m,l-1)}$ contain the features of the noised image after $l-1$ blocks, and $\boldsymbol{c}_{\text{text}}$, $\boldsymbol{C}$ denote the text and pixel-level controls, respectively. Considering the case $\boldsymbol{C} := \{\boldsymbol{c}^{(1)}\}$ which is consistent with past work [14], the pixel-level control is incorporated via:

$$\tilde{f}^{(l)}\left(\boldsymbol{x}^{(m,l-1)}, m, \boldsymbol{c}_{\text{text}}, \boldsymbol{C}\right) :=$$

$$\mathcal{Z}_{\text{out}}\left(f^{(l)}\left(\boldsymbol{x}^{(m,l-1)} + \mathcal{Z}_{\text{in}}\left(\boldsymbol{c}^{(1)}\right), m, \boldsymbol{c}_{\text{text}}\right)\right), \quad (11)$$

where $\mathcal{Z}_{\text{in}}$ and $\mathcal{Z}_{\text{out}}$ are the newly attached zero convolution layers, and $f^{(l)}$ is initialized from the l^{th} encoder block of the pretrained text-conditioned UNet.

In Music ControlNet, we revamp the control process for multiple time-varying controls to be:

$$\tilde{f}^{(l)}\left(\boldsymbol{x}^{(m,l-1)}, m, \boldsymbol{c}_{\text{text}}, \boldsymbol{C}\right) :=$$

$$\mathcal{Z}_{\text{out}}\left(f^{(l)}\left(\boldsymbol{x}^{(m,l-1)} + \mathcal{Z}_{\text{in}}\left(\mathcal{M}^{(n)}(\boldsymbol{c}^{(n)})\right), m, \boldsymbol{c}_{\text{text}}\right)\right), \quad (12)$$

where $\mathcal{M}^{(n)}$ is an additional 1-hidden-layer MLP that transforms B_n for the nth control signal, the number of classes for the control $\boldsymbol{c}^{(n)}$ following (10), to match the number of frequency bins B, and simultaneously learns the relationship between control classes and frequency bins. In cases with multiple controls, i.e., $\boldsymbol{C} = \{\boldsymbol{c}^{(n)}\}_{n=1}^{N}$, each control is processed with its individual MLP, i.e., $\mathcal{M}^{(n)}$, and then concatenated along the depth dimension, i.e., D_n, before entering the shared zero-convolution layer $\mathcal{Z}_{\text{in}}$. For the case of one control signal and no MLP adaptor, (12) reduces to (11) and past work.

C. Masking Strategy to Enable Partially-Specified Controls

To give creators the freedom to input any subset of the N controls, Uni-ControlNet [15] proposed a CFG-like training strategy to drop out each of the control signals $\boldsymbol{c}^{(n)}$ randomly during training. We follow the same strategy and further assign a higher probability to keep or drop all controls [15] as we found that this leads to perceptually better generations. In more detail, we let the index set of control signals be $\mathcal{I} = \{1, \ldots, N\}$. At each training step, we then select a subset $\mathcal{I}' \subseteq \mathcal{I}$ that will be set to zero or dropped. We then directly apply the index subset to the control signals via:

$$\boldsymbol{c}^{(n)} := \begin{cases} \boldsymbol{0}_{\text{Tf}_\text{k} \times B_n \times D_n} & \forall n \in \mathcal{I}' \\ \boldsymbol{c}^{(n)} & \forall n \in \mathcal{I} \setminus \mathcal{I}'. \end{cases} \quad (13)$$

Doing so induces $f_\theta(\boldsymbol{x}^{(m)}, m, \boldsymbol{c}_{\text{text}}, \boldsymbol{C})$ to learn the correspondence between any subset of controls and the outputs.

In Music ControlNet, we further desire a model that allows the given subset of controls to be *partially-specified in time*. Therefore, we devise a new scheme that partially masks active controls (i.e., those indexed by $\mathcal{I} \setminus \mathcal{I}'$). Specifically, we randomly sample

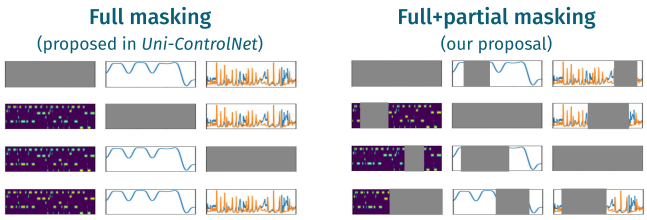

Fig. 2. Two masking schemes we randomly choose to apply during training that allow creators to input *any subset* of the time-varying controls, *fully* or *partially* specified in time, at inference. Each row indicates a unique masking instantiation over the set of control signals $\boldsymbol{C} := \{\boldsymbol{c}^{(n)}\}_{n=1}^{N}$ ($N = 3$ as illustrated here). Masked control signals are colored in gray. Within each scheme, we show the melody (left), dynamics (middle), and rhythm (right) control signals.

a pair $(t_{n,a}, t_{n,b}) \in \{1, \ldots, \mathrm{Tf_k}\}^2$, where $t_{n,a} < t_{n,b}$, for each of the active controls, and mask them as:

$$\boldsymbol{c}_t^{(n)} := \begin{cases} \boldsymbol{0}_{B_n \times D_n} & \text{if } t \in [t_{n,a}, t_{n,b}] \\ \boldsymbol{c}_t^{(n)} & \text{otherwise.} \end{cases} \quad \forall n \in \mathcal{I} \setminus \mathcal{I}' \qquad (14)$$

Fig. 2 displays example instantiations of the two masking schemes detailed above. At each training step, after selecting $\mathcal{I}'$ (i.e., determining the dropped controls) we choose one of the two masking schemes uniformly at random, and then sample the timestamp pairs (i.e., $(t_{n,a}, t_{n,b})$'s) when needed. In this way, we further employ a CFG-like training strategy to enable partially-specified controls in a unified manner.

D. Musical Control Signals

We propose three control signals, i.e., $\boldsymbol{C}$, we believe are useful for creators including *melody*, *dynamics*, and *rhythm*. Furthermore, we define two methods for obtaining control signals: *extracted controls* and *created controls*. Extracted controls are signals extracted from an input audio example or the target spectrogram, i.e, $\boldsymbol{s} \in \mathbb{R}^{\mathrm{Tf_k} \times B \times D}$ from a feature extraction function, enabling style transfer applications by example, and requiring no human annotation. Created controls are signals directly annotated, modified, or otherwise created from a music creator at inference-time to compose their music from scratch. We train our method on extracted controls and run inference on either extracted or created controls. Below, we introduce how our controls are obtained, what connections they have to music, and how creators can form created controls at inference-time. Readers may refer to Fig. 3 for how the control signals may be visually presented to creators.

- **Melody ($\boldsymbol{c}_{\mathbf{mel}} \in \mathbb{R}^{\mathrm{Tf_k} \times 12 \times 1}$):** Following [5], we adopt a variation of the chromagram [26] to encode the most prominent musical tone over time. To do so, we compute a linear spectrogram and then rearrange the energy across the B frequency bins into 12 pitch classes (or semitones, i.e., C, C-sharp,..., B-flat, B) in a frame-wise manner, i.e., independently for each $t \in \{1, \ldots, \mathrm{Tf_k}\}$, via the Librosa Chroma function [27]. To form a better proxy for melody from the raw chromagram, only the most prominent pitch class is preserved by applying an argmax operation to make the chromagram frame-wise one-hot. Additionally, we apply a Biquadratic high-pass filter [28] with a cut-off at Middle C, or 261.2 Hz) before chromagram computation to avoid bass dominance, i.e., the resulting one-hot chromagram encodes the bass notes, rather than the desired melody notes. At test time, the melody control can be created by recording a simple melody, or simply drawing the pitch contour. A desirable model should be able to turn the simple created melody control into rich, high-quality multitrack music.
- **Dynamics ($\boldsymbol{c}_{\mathbf{dyn}} \in \mathbb{R}^{\mathrm{Tf_k} \times 1 \times 1}$):** The dynamics control is obtained by summing the energy across frequency bins per time frame of a linear spectrogram, and mapping the resulting values to the decibel (dB) scale, which is closely linked to loudness perceived by humans [27]. To mitigate rapid fluctuations of the raw dynamic values due to note or percussion onsets, and also to bring our dynamics control closer to the perceived musical intensity, we apply a smoothing filter with one second context window over the frame-wise values (i.e., a Savitzky-Golay filter [29]). The dynamics control not only characterizes the loudness of notes, but also is strongly correlated with important musical intensity-related attributes like instrumentation, harmonic texture, and rhythmic density thanks to the natural correlation between loudness and the aforementioned attributes in human-composed music. During inference, creators can simply draw a line/curve of how they want the musical intensity to vary over time as the created dynamics control.
- **Rhythm ($\boldsymbol{c}_{\mathbf{rhy}} \in \mathbb{R}^{\mathrm{Tf_k} \times 2 \times 1}$):** For rhythm control, we employ an in-house implementation of an RNN-based beat detector [30] that is trained on a different internal dataset to predict whether a frame is situated on a beat, a downbeat, or neither. We then use the frame-wise *beat* and *downbeat* probabilities for control, resulting in 2 classes per frame. The advantages of our time-varying beat/downbeat control over just inputting a global tempo (i.e, beats per minute) are: (i) it allows creators to precisely synchronize beats/downbeats with, for example, video scene cuts or other moments of interest in the content to be paired with generated music. (ii) it encodes some nuanced information of rhythmic feeling, e.g., whether the music sounds more harmonic or rhythmic, and whether the rhythmic pattern is clear/simple, or complex, on which experienced music creators may want to influence in the generative process. At inference, the rhythm control can be created by time-stretching the beat/downbeat probability curves extracted from existing songs to match the desired tempo. Also, creators can obtain precise beat/downbeat timestamps by feeding the beat/downbeat curves to a Hidden Markov Model (HMM) based post-filter [31], [32], and use the timestamps to shift the curves along the time axis for synchronization purposes mentioned above. We also tried to manually draw spiked curves as the created rhythm control, but the performance of this was worse than our final hand-drawn (i.e., created) dynamics control.

IV. Experimental Setup

A. Datasets

We train our models on a dataset of ≈1800 hours of licensed instrumental music with genre and mood tags. Our dataset does not have free-form text description, so we use class-conditional

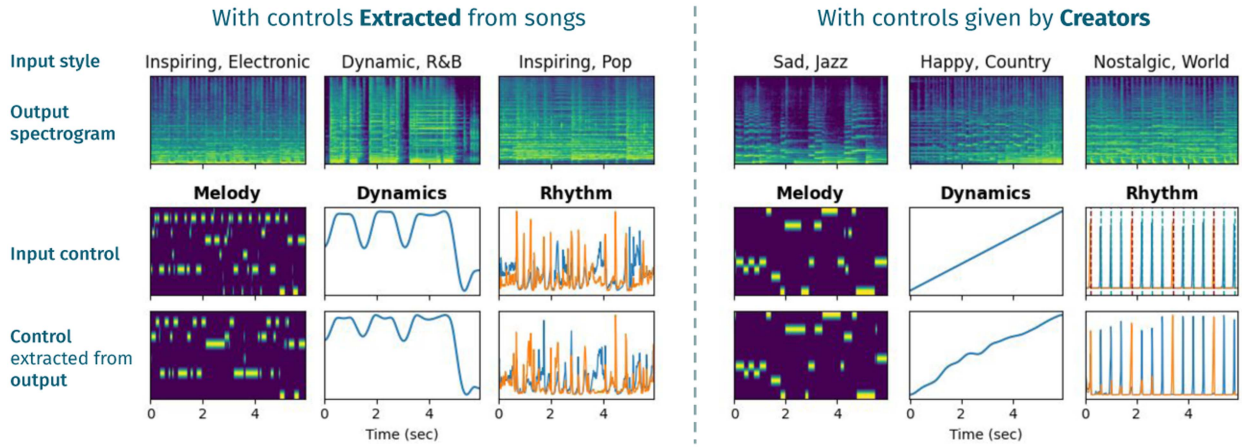

Fig. 3. Examples of Music ControlNet generations given single time-varying controls. Our model faithfully follows all controls despite their non-obvious relationship with the spectrograms. Controls given by creators may be, e.g., simple melodies, drawn dynamics curves, and time-shifted/stretched rhythm templates as shown here. (Colors in rhythm control represent beat/downbeat probabilities. Dashed lines in the creator rhythm control are beat/downbeat timestamps that can be used to sync beats as desired.)

text control of global musical style, as done in JukeBox [1]. For evaluation, we use data from four sources:

1) an **in-domain test set** with 2 K songs held out from our dataset,
2) the **MusicCaps dataset** [2] with around 5 K 10-second clips associated with free-form text description
3) the **MusicCaps+ChatGPT dataset** where we use ChatGPT [33] to convert the free-form text in MusicCaps to mood and genre tags that match our dataset via the prompt "*For the lines of text below, convert each to one of the following [genres or moods] and only output the [genre or mood] per line (no bullets): [MusicCaps description]*", and
4) a **Created Controls dataset** of control signals that music creators can realistically give via manual annotation or similar.

B. Created Controls Dataset Details

For our Created Controls dataset, we created example melodies, dynamics annotations, and rhythm presets that we envision creators would use during music co-creation via:

- **Melody:** We record our piano play of 10 well-known classical public domain music melodies (30 seconds long each) composed by Bach, Vivaldi, Mozart, Beethoven, Schubert, Mendelssohn, Bizet, and Tchaikovsky, and crop two 6-second chunks, minimizing repeated musical content as possible, resulting in a 20-example melody controls.
- **Dynamics:** To simulate a creator-drawn dynamics curves, we draw out 6-second long dynamics curves as $\{\text{Linear}, \text{Tanh}, \text{Cosine}\}$ functions, either vertically flipped or not, with scaled dynamics ranges of $\{\pm 6, \pm 9, \pm 12, \pm 15\}$ decibels from the mean value of all training examples. This leads to $3 \times 2 \times 4 = 24$ created dynamics controls.
- **Rhythm:** We create "rhythm presets" via selecting four songs from our **in-domain test set** with different rhythmic strengths and feelings, extract their rhythm control signals, and time-stretch them using PyTorch interpolation with factors $\{0.8, 0.9, 1.0, 1.1, 1.2\}$ to create 20 rhythm controls.

Each set of created controls is then cross-producted with 10 genres $\times$ 10 moods to form the final dataset of 2.0 K, 2.4 K and 2.0 K samples. Our created controls are distinct from controls that are directly extracted from mixture data during training.

C. Model, Training, and Inference Specifics

For our spectrogram generation model $p_\theta(\boldsymbol{s} \mid \boldsymbol{c}_{\text{text}}, \boldsymbol{C})$, we use a convolutional UNet [21] with 5 2D-convolution ResNet [34] blocks with $[64, 64, 128, 128, 256]$ feature channels per block with a stride of 2 in between downsampling blocks. The UNet inputs Mel-scaled [35] spectrograms clipped to a dynamic range of 160 dB and scaled to $[-1, 1]$ computed from 22.05 kHz audio with a hop size of 256 (i.e., frame rate $\text{f}_\text{k} \approx 86$ Hz), a window size of 2048, and 160 Mel bins. For our genre and mood global style control $\boldsymbol{c}_{\text{text}}$, we use learnable class-conditional embeddings with dimension of 256 that are injected into the inner two ResNet blocks of the U-Net via cross-attention. We use a cosine noise schedule with 1000 diffusion steps m that are injected via sinusoidal embeddings with a learnable linear transformation summed directly with U-Net features in each block. To approximately match the output dimensions of ControlNet ($512 \times 512 \times 3$), we set our output time dimension to 512 or $\approx$6 seconds, yielding a $512 \times 160 \times 1$ output dimension. We use an L1 training objective between predicted and actual added noise, an Adam optimizer with learning rate to 10^{-5} with linear warm-up and cosine decay. Due to limited data and efficiency considerations, we instantiate a relatively small model of 41 million parameters and pretrain with distributed data parallel for 5 days on 32 A100 GPUs with a batch size of 24 per GPU.

Given our pretrained global style control model, we finetune on time-varying melody, dynamics, and rhythm controls controls. The time-varying controls enter the pretrained U-Net via

an adaptor branch as discussed above. We use the same loss and optimizer used for pretraining and finetune until convergence for 3 days with 8 A100 GPUs. At inference, we use 100-step DDIM [20] sampling, and CFG [24] on global style control with a scale of 4 on the global style control only.

For our spectrogram-to-audio vocoder $p_\phi(\boldsymbol{w} \mid \boldsymbol{s})$, we train a diffusion-based DiffWave [25] vocoder for our main results (code available) and use the MusicHiFi vocoder [36] for our demo video. We leverage an open-source package [37], and use our main training dataset, an Adam optimizer with learning rate of 10^{-5}, noise prediction L1 loss, a 50-step linear noise schedule, hopsize of 256 samples, sampling rate of 22050 Hz, batch size of 50 per GPU and train on 8 GPUs for 10 days. For inference, we use DDIM-like sampling [25] with six steps.

D. Evaluation Metrics

We evaluate time-varying controllability, adherence to global text control, and audio realism via the metrics below.

- **Melody accuracy** examines whether the frame-wise pitch classes (C, C#,..., B; 12 in total) match between the input melody control and that extracted from the generation.
- **Dynamics correlation** is the Pearson's correlation between the frame-wise input dynamics values to the values computed from the generation. We compute two types of correlation, which we call *micro* and *macro* correlation respectively. *Micro* computes r's separately for each generation, while *macro* collects input/generation dynamics values from all generations, and then computes a single r. The *micro* correlation examines whether relative dynamics control values *within a generation* is respected, while the *macro* one checks the same property *across many generations*.
- **Rhythm F1** follows the standard evaluation methodology for beat/downbeat detection [38], [39]. It quantifies the alignment between the beat/downbeat timestamps estimated from the input rhythm control, and those from the generation. The timestamps are estimated by applying an HMM post-filter [31] on the frame-wise (down)beat probabilities (i.e., the rhythm control signal). Following [39], a pair of input and generated (down)beat timestamps are considered aligned if they differ by < 70 milliseconds.
- **CLAP** score [40], [41] evaluate text control adherence via computing the pair-wise cosine similarity of text and audio embeddings extracted from CLAP. CLAP is a dual-encoder foundation model where the encoders respectively receive a text input and an audio input. The text and audio embedding spaces are learned via a contrastive objective [42]. To obtain the embeddings for evaluation, we feed the generated audio to the CLAP audio encoder, and set the CLAP text encoder input to "*An audio of [mood] [genre] music*" to accommodate our tag-based control on global musical style.
- **FAD** is the Fréchet distance between the distribution of embeddings from a set of reference audios and that from generated audios [43]. It measures '*how realistic the set of generated audios are*', taking both quality and diversity into account. To ensure comparable FAD scores, we utilize the Google Research FAD package [43], which employs a VGGish [44] model trained on audio classification [45] to extract embeddings from audios. Unless otherwise specified, the reference audios for FAD are our in-domain test dataset.

V. Evaluation and Discussion

We conduct a comprehensive evaluation of our proposed Music ControlNet framework. Specifically, we perform quantitative studies of (i) single vs. multiple time-varying extracted controls, (ii) extracted controls vs. created controls, (iii) fully vs. partially-specified created controls, (iv) extrapolating generation duration beyond the training duration (i.e., 6 seconds), and (v) benchmarking with the 1.5 billion-parameter MusicGen model with melody control. In all experiments, a single fine-tuned model is used with different inference configurations. The duration of generation is 12 or 24 seconds in experiment (iv), 10 seconds in experiment (v) so we can be consistent with MusicCaps [2] benchmark, and 6 seconds in all other experiments. We leverage the fully convolutional nature of our UNet backbone to generate music that is longer than what is seen during training. We conclude with an in-depth qualitative analysis of created generation examples.

A. Single & Multiple Extracted Controls

We evaluate generation performance by applying different combinations of controls at inference time, using single or multiple control signals *extracted* from our **in-domain test set**. The results are shown in Table I. First, we compare generations using global style only (i.e., genre and mood tags) and those with single time-varying controls (rows 2∼4). When the corresponding controls are enforced, we observe much higher melody accuracy, dynamics correlations, and rhythm F1s, which indicate that our proposed control injection mechanism (see Section III-B) affords effective time-varying controllability. Interestingly, we find in rows 3 and 4 that the dynamics and rhythm metrics are higher compared to using global style control only (1st row) even when the corresponding controls are excluded. We hypothesize that this is due to that our rhythm and dynamics controls have natural correlation.

Second, focusing on generations with multiple controls (last 4 rows in Table I), we find the time-varying controllability metrics to remain largely the same compared to single control scenarios. This shows that our model learns to simultaneously respond to multiple controls well despite the added complexity. However, as more time-varying controls are enforced, text control adherence (CLAP score) degrades mildly, while overall audio realisticness (FAD) is not negatively impacted.

B. From Extracted to Created Controls

To empower creators to generate music with their own ideas, we evaluate the single-control generations using *created* controls from our **Created Controls dataset**. The comparison with extracted controls are displayed in Table II. We notice several

TABLE I
PERFORMANCE OF SINGLE VS. MULTIPLE TIME-VARYING CONTROLS USING CONTROLS *EXTRACTED* FROM OUR IN-DOMAIN TEST SET

	Control signals mel	dyn	rhy	**Melody** acc (%)	**Dynamics** corr (r, in %) Micro	Macro	**Rhythm** F1 (%) Beat	Downbeat	**CLAP**	**FAD** ↓
Global style only	✗	✗	✗	8.5	−0.7	0.7	27.8	7.8	0.28	1.51
Single controls	✓	✗	✗	**58.3**	4.4	3.1	40.2	12.1	0.28	1.34
	✗	✓	✗	8.6	**88.8**	**63.6**	36.7	16.1	0.26	1.50
	✗	✗	✓	8.6	25.8	34.6	**69.2**	**35.4**	0.27	1.17
Multi controls	✓	✓	✗	**57.7**	**89.7**	**64.8**	47.4	21.8	0.26	1.38
	✓	✗	✓	**59.1**	31.6	36.3	**70.0**	**38.7**	0.26	1.16
	✗	✓	✓	8.7	**89.6**	**60.9**	**72.1**	**39.9**	0.26	1.12
	✓	✓	✓	**58.7**	**90.8**	**64.0**	**70.8**	**40.8**	0.25	1.14

Timevarying controllability (i.e., Melody, Dynamics, Rhythm metrics) is notably higher when the corresponding control is passed to the model (✓) as opposed to being excluded (✗). Bold values indicates enforced controls. Higher is better except for FAD.

TABLE II
EVALUATION ON CONTROLS *CREATED* BY CREATORS THAT ARE MORE SIMPLE THAN THE EXTRACTED CONTROLS SEEN BY OUR MODEL DURING TRAINING

Control	Control source	**Melody** acc (%)	**Dynamics** corr (r, in %) Micro	Macro	**Rhythm** F1 (%) Beat	Downbeat	**CLAP**	**FAD** ↓
Melody	**Extracted**	58.3	—	—	—	—	0.28	1.34
	Created	78.2	—	—	—	—	0.27	1.81
Dynamics	**Extracted**	**—**	88.8	63.6	—	—	0.26	1.50
	Created	**—**	98.5	93.2	—	—	0.26	2.18
Rhythm	**Extracted**	**—**	—	—	69.2	35.4	0.26	1.17
	Created	**—**	—	—	88.6	45.2	0.26	2.93

Using created controls leads to better time-varying controllability, i.e., Melody, Dynamics, Rhythm metrics.

TABLE III
EVALUATION ON CONTROLS *PARTIALLY SPECIFIED* IN TIME, WHICH LIFT THE REQUIREMENT FOR CREATORS TO ALWAYS INPUT FULL CONTROLS

Control	Control source & span	**Melody** acc (%)	**Dynamics** corr (r, in %) Micro	Macro	**Rhythm** F1 (%) Beat	Downbeat	**CLAP**	**FAD** ↓
Melody	**Created, full**	78.2	—	—	—	—	0.27	1.81
	Created, partial	74.3	—	—	—	—	0.27	1.66
Dynamics	**Created, full**	**—**	98.5	93.2	—	—	0.26	2.18
	Created, partial	**—**	88.6	89.0	—	—	0.27	1.52
Rhythm	**Created, full**	**—**	—	—	88.6	45.2	0.26	2.93
	Created, partial	**—**	—	—	80.1	34.8	0.26	2.60

Time-varying contrallability (Melody, Dynamics, Rhythm metrics) only degrades mildly with partially-specified controls.

interesting insights. First and perhaps unexpectedly, we find that across all three control signals, all time-varying controllability metrics actually improve when using created controls. This demonstrates our model's generalizability to out-of-domain control inputs.

Second, we find that global style control adherence (CLAP score) is largely unaffected, while FAD appears to degrade. The degradation in FAD is multifaceted. On the one hand, the created controls, naturally creates some music that is distributionally different from the in-domain test set. Hence, we can not expect the desirable generations to score a low FAD. On the other hand, perceptually, we do find the generations with created controls are more often less musically interesting. We find this true particularly for created melody and dynamics controls, where the model may copy the melody with a single instrument on a constant background chord, or match dynamics using monotonous bass or sound effects. However, in practice, we believe this is not an issue as creators can ask for a batch of generations and select the best one.

C. From Fully- to Partially-Specified Controls

We evaluate generation quality using partially-specified, created control signals (made possible by the masking scheme in Section III-C) and compare fully-specified created controls in Table III. For partially-specified cases, for each sample, we specify the control for a random 1.0 to 4.5-second span out of the full 6-second duration. The melody, dynamics, rhythm metrics are computed only within the partially-specified spans, while CLAP and FAD still take the full generated audio as input. Overall, we find that partial control somewhat degrades time-varying controllability compared to the full created control scenarios, but it remains strong and mostly better than using full extracted controls (cf. rows marked by **Extracted** in Table II). Global style control adherence (CLAP) is unaffected. Overall quality (FAD) improves, suggesting that the less amount of controls induces the generations to match the training distribution better. We also found that the coexistence of controlled and uncontrolled spans did not lead to pronounced incoherence issues.

TABLE IV
EVALUATION OF GENERATIONS OF LONGER DURATIONS THAN THAT SEEN AT TRAINING (I.E., 6 SEC), USING *CREATED* MELODIES

Length	**Melody** acc(%)	**CLAP**	**FAD** ↓
6 sec	78.2	0.27	1.81
12 sec	81.0	0.32	2.11
24 sec	82.8	0.33	2.54

D. Extrapolating Duration of Generation

The 6-second duration of our model can be restrictive for some real-world use cases. Therefore, we capitalize on the inherent length-extrapolation ability of our fully convolutional model backbone, and experiment with 12 and 24 s-long generations (i.e., 2x and 4x the duration at training) using created melody controls. The evaluation results are in Table IV. We observe that both time-varying controllability and text control adherence are retained, but the overall audio realisticness, measured by FAD, somewhat degrades. We verify this degradation via listening and note that the background noise level noticeably increases as we extrapolate duration.

E. Benchmarking With MusicGen on Melody Control

We compare our model trained with melody, dynamics, and rhythm controls to the 1.5B-parameter MusicGen [5] model trained with melody and free-form text control. We use the MusicGen model in three scenarios:

1) *text-only generation*, where we do not pass in melodies,
2) *full melody control*, where we pass in melodies that are as long as generation length, and
3) *1/2 prompt melody control*, where the melodies passed in are half length.

For our model, we achieve these scenarios via omitted, partially-specified, or full melody control.

As MusicGen support free-form text control and arbitrary generated audio length, we use both the **MusicCaps** and **MusicCaps+ChatGPT** datasets. Both datasets contains the same audio, but the **MusicCaps+ChatGPT** dataset has the text descriptions converted into genre & mood tags by ChatGPT. The ChatGPT-converted tags are then used in two ways: as the global style input to our model, and as text input when computing CLAP scores. That is, we have two versions of CLAP when comparing our model to MusicGen, namely, $\textbf{CLAP}_{\text{text}}$, which measures CLAP with (original free text, generation audio) tuples, and $\textbf{CLAP}_{\text{tag}}$ (i.e., the CLAP metric used in previous experiments), which only allows converted tags as text input to both MusicGen (written as text, e.g., "*An audio of happy jazz music*") and our model, and measures CLAP with (converted tags, generation audio) tuples. We also compute two versions of FAD scores, one using MusicCaps as the reference set (i.e., $\textbf{FAD}_{\text{MCaps}}$) and the other using our in-domain test set as the reference (i.e., $\textbf{FAD}_{\text{Ours}}$). We generate 10-second long outputs to be consistent with the MusicCaps dataset and evaluation protocol.

We consider both *extracted* and *created* melody controls in this comparison. As shown in Table V, we find our proposed work responds more precisely to the melody control, particularly on created melodies, where our model is as much as 49% relatively more faithful to the control. In terms of text control adherence, when the text input is restricted to the converted mood & genre tags (i.e., the $\textbf{CLAP}_{\text{tag}}$ metric), our model is comparable to MusicGen. On overall audio realisticness, as our model is much smaller than MusicGen, and trained on a much more restricted domain of data, it is unsurprising that it scores a worse FAD when using MusicCaps recordings. Moreover, we note that many examples in the MusicCaps datast are, in fact, low-quality audio recordings and/or contain vocals which our model never sees during training, which may render $\textbf{FAD}_{\text{MCaps}}$ biased against our model. We also note when the reference set is our in-domain test set audios (i.e., $\textbf{FAD}_{\text{ours}}$), we are competitive to or somewhat better than MusicGen. Finally, we note MusicGen can generate unlimited length music via an auto-regressive architecture, while our approach cannot.

F. Qualitative Analysis of Generations

In Fig. 3, we show generation outputs with each of the proposed controls, i.e., melody, dynamics, or rhythm, either *extracted* or *created*. Concentrating first on the extracted controls (Fig. 3, left half), all of the three control signals are closely followed by our model even with their different dimensions and relationships w.r.t. the spectrogram. Moving on to the created controls (Fig. 3, right half), the controls are almost perfectly reflected despite some of them (i.e., melody & dynamics) being out-of-domain from training data. Moreover, our approach is able to wield musical creativity even though the created controls are much simpler than extracted ones. For example, visible from the output spectrograms given melody or dynamics controls, our model generates music with varying texture and rhythmic patterns, rather than simply replicating the monophonic melody, or changing the volume of a single note to match the increasing dynamics.

Fig. 4 displays generations using multiple created controls, specifically, with *a)* full melody & dynamics controls simultaneously enforced, and *b)* all three controls with partially-specified spans, which simulate the creator intent: "*I want the music to start with my signature melody, and have it intensifying at the end with beats synchronized to my video scene cuts to engage my audience.*" Example *a)* verifies the composability of created controls (as opposed to extracted ones, which has been examined in Table I) as both controls are respected by the model. Example *b)* demonstrates effective control even when controls signals are partially specified, and the capability to generate cohesive music (i.e., the output spectrogram contains no visible borders) when both controlled and uncontrolled spans are present.

VI. RELATED WORK

A. Text-to-Music Generation

Music ControlNet builds on a recent body of work on text-to-music, where the goal is to generate music audio conditioned on text descriptions or categories [1], [2], [3], [4], [46], [47]. This line of research is bifurcated into two broad methodological

TABLE V
COMPARISON TO MUSICGEN [5] ON THE MUSICCAPS DATASET [2]

Control	Model	**Extracted** melody control					**Created** melody control				
		Melody	**CLAP**$_{\text{tag}}$	**CLAP**$_{\text{text}}$	**FAD**$_{\text{MCaps}}$ ↓	**FAD**$_{\text{ours}}$ ↓	**Melody**	**CLAP**$_{\text{tag}}$	**CLAP**$_{\text{text}}$	**FAD**$_{\text{MCaps}}$ ↓	**FAD**$_{\text{ours}}$ ↓
Text only	Ours	—	**0.33**	0.20	10.5	**2.5**	—	—	—	—	—
	MusicGen	—	0.32	**0.28**	**4.6**	3.8	—	—	—	—	—
Melody (full)	Ours	**47.1**	0.33	0.22	10.8	**2.5**	**82.6**	0.33	0.19	11.2	**2.0**
	MusicGen	41.3	**0.34**	**0.29**	**5.7**	**2.5**	55.2	**0.34**	**0.28**	**6.2**	2.8
Melody (½ prompt)	Ours	**46.7**	0.33	0.21	10.9	2.5	**80.8**	0.33	0.20	11.1	**1.9**
	MusicGen	42.1	**0.34**	**0.29**	**5.7**	**2.3**	56.8	**0.34**	**0.28**	**6.1**	2.3

Input melodies are either extracted from MusicCaps recordings or randomly selected from 20 melodies from our created controls dataset. Our model exhibits more precise melody control, especially on created melodies, with comparable text control adherence when restricting MusicGen text prompts to our dataset's mood and genre tags (i.e., CLAP$_{\text{tag}}$). Note that our model (41M parameters) is much smaller than MusicGen (1.5B parameters), and additionally accepts multiple controls and partially-specified spans.

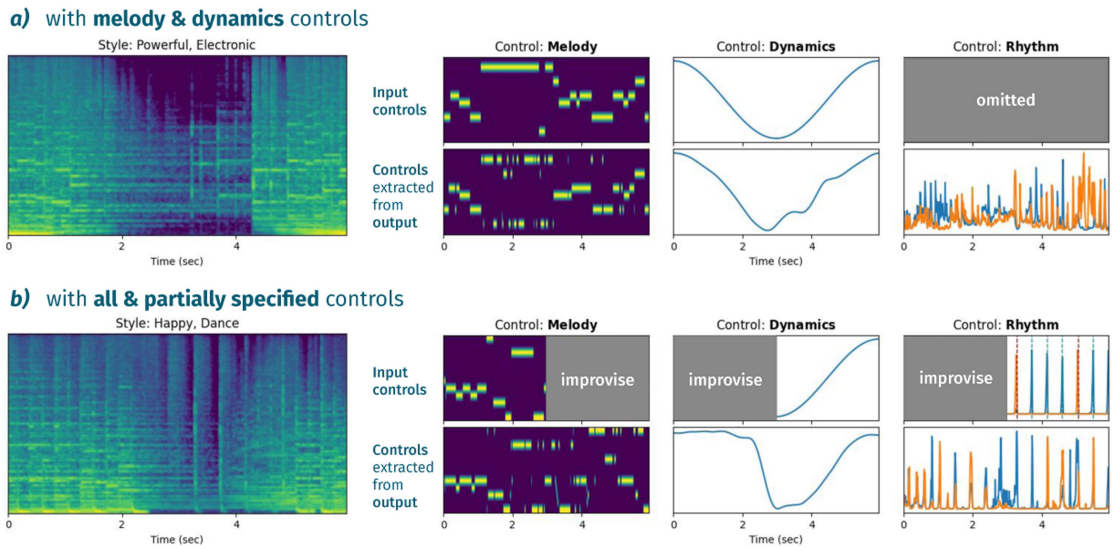

Fig. 4. Music ControlNet generations with multiple and/or partially-specified controls given by *creators*. All controls are honored when enforced, demonstrating the composability of our controls. In uncontrolled segments, the generations exhibits consistent style and musical creativity.

branches which build on advances in natural language processing and computer vision respectively:

1) using LLMs to model tokens from learned audio codecs as proposed in [48], [49], and
2) using (latent) diffusion to model image-like spectrograms. We explore diffusion to leverage strong inductive biases developed for spatial control.

B. Time-Varying Controls for Music Generation

Our approach is related to generating music audio from time-varying control. A contemporaneous work is [50], which focuses on a similar goal to ours, but is built on pretrained large language models (LLMs) instead of diffusion models. Work on style transfer includes methods to convert musical recordings in one style to another while preserving underlying symbolic music [8], [9], [10], [11]. Other work explores directly synthesizing symbolic music (e.g., MIDI) into audio [6], [7]. Both approaches require training individual models per style rather than leveraging text control for style, and needs complete musical inputs rather than simpler controls we explore here. More recently, [16], [51], [52] generate music in broad styles with time-varying control but target tasks with stronger conditions like musical accompaniment or variation generation, which are different applications than ours. Another body of research [53], [54], [55], [56] explores time-varying controls for symbolic-domain music generation, i.e., modeling sheet music or MIDI events. The controls considered in these works are of coarser time scale, e.g., at the measure or phrase level, while our approach offers precise control down to the frame level.

C. Unconditional Music Generation

Our work on controllable music audio generation builds on earlier work on unconditional generative modeling of audio. Early approaches explored directly modeling audio waveforms [57], [58], [59]. More recent work [48], [49], [60], [61] favors hierarchical approaches like those we consider there.

VII. CONCLUSION

We proposed Music ControlNet, a framework that enables creators to harness music generation with precise, multiple time-varying controls. We demonstrate our framework via melody,

dynamics, and rhythm control signals, which are all basic elements in music and complement with each other well. We find that our framework and control signals not only enables any combination of controls, fully- or partially-specified in time, but also generalizes well to controls we envision creators would employ.

Our work paves a number of promising avenues for future research. First, beyond melody, dynamics, and rhythm controls, several additional musical features could be employed such as chord estimation for harmony control, multi-track pitch transcription, instrument classification, or even more abstract controls like emotion and tension. Second, as the set of musical controls becomes large, generating control presets based on text, speech, or video inputs could make controllable music generation systems more approachable to a wide range of content creators. Last but not least, addressing the domain gap between extracted and created controls via, e.g., adversarial approaches [62], could further enhance the musical quality of generations under created controls.

VIII. Ethics Statement

Music generation is poised to upend longstanding norms around how music is created and by whom. On the one hand, this presents an opportunity to increase the accessibility of musical expression, but on the other hand, existing musicians may be forced to compete against generated music. While we acknowledge our work carries some risk, we sharply focus on improving control methods so as to directly offer musicians more creative agency during the generation process. Other potential risks surround the inclusion of singing voice, accidental imitation of artists without their consent, and other unforeseen ethical issues, so we use licensed instrumental music for training and melodies extracted from our training data or public domain melodies we recorded ourselves for inference. For evaluation, we do use the MusicCaps dataset [2] as it is standard in recent text-to-music generation literature.

References

[1] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, "Jukebox: A generative model for music," 2020, *arXiv:2005.00341*.

[2] A. Agostinelli et al., "MusicLM: Generating music from text," 2023, *arXiv:2301.11325*.

[3] Q. Huang et al., "Noise2Music: Text-conditioned music generation with diffusion models," 2023, *arXiv:2302.03917*.

[4] H. Liu et al., "Audio LDM: Text-to-audio generation with latent diffusion models," in *Proc. Int. Conf. Mach. Learn.*, 2023.

[5] J. Copet et al., "Simple and controllable music generation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2023.

[6] C. Hawthorne et al., "Enabling factorized piano music modeling and generation with the MAESTRO dataset," in *Proc. Int. Conf. Learn. Representations*, 2019.

[7] C. Hawthorne et al., "Multi-instrument music synthesis with spectrogram diffusion," in *Proc. ISMIR*, 2022.

[8] N. Mor, L. Wolf, A. Polyak, and Y. Taigman, "A universal music translation network," in *Proc. Int. Conf. Learn. Representations*, 2019.

[9] S. Huang, Q. Li, C. Anil, X. Bao, S. Oore, and R. B. Grosse, "TimbreTron: A WaveNet(CycleGAN(CQT(Audio))) pipeline for musical timbre transfer," in *Proc. Int. Conf. Learn. Representations*, 2019.

[10] J. Engel, L. Hantrakul, C. Gu, and A. Roberts, "DDSP: Differentiable digital signal processing," in *Proc. Int. Conf. Learn. Representations*, 2020.

[11] A. Caillon and P. Esling, "RAVE: A variational autoencoder for fast and high-quality neural audio synthesis," 2021, *arXiv:2111.05011*.

[12] Y. Wu et al., "MIDI-DDSP: Detailed control of musical performance via hierarchical modeling," in *Proc. Int. Conf. Learn. Representations*, 2022.

[13] C. J. Steinmetz, N. J. Bryan, and J. D. Reiss, "Style transfer of audio effects with differentiable signal processing," *J. Audio Eng. Soc.*, vol. 70, pp. 708–721, 2022.

[14] L. Zhang, A. Rao, and M. Agrawala, "Adding conditional control to text-to-image diffusion models," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 3836–3847.

[15] S. Zhao et al., "Uni-ControlNet: All-in-one control to text-to-image diffusion models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2023.

[16] C. Donahue et al., "SingSong: Generating musical accompaniments from singing," 2023, *arXiv:2301.12662*.

[17] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 2256–2265.

[18] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 6840–6851.

[19] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proc. Int. Conf. Learn. Representations*, 2013.

[20] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," in *Proc. Int. Conf. Learn. Representations*, 2020.

[21] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Interv.*, 2015, pp. 234–241.

[22] C. Saharia et al., "Photorealistic text-to-image diffusion models with deep language understanding," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 36479–36494.

[23] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 10684–10695.

[24] J. Ho and T. Salimans, "Classifier-free diffusion guidance," in *Proc. NeurIPS Workshop Deep Gen. Models Downstream Appl.*, 2021.

[25] Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro, "DiffWave: A versatile diffusion model for audio synthesis," in *Proc. Int. Conf. Learn. Representations*, 2021.

[26] M. Müller, *Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications*. Berlin, Germany: Springer, 2015.

[27] B. McFee et al., "librosa/librosa: 0.10.1," 2023, doi: 10.5281/zenodo.8252662.

[28] Y.-Y. Yang, "Torchaudio: Building blocks for audio and speech processing," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2021, pp. 6982–6986.

[29] P. Virtanen et al., "SciPy 1.0: Fundamental algorithms for scientific computing in python," *Nature Methods*, vol. 17, no. 3, pp. 261–272, 2020.

[30] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer, "Madmom: A new python audio and music signal processing library," in *Proc. ACM Multimedia*, 2016, pp. 1174–1178.

[31] F. Krebs, S. Böck, and G. Widmer, "An efficient state-space model for joint tempo and meter tracking," in *Proc. ISMIR*, 2015, pp. 72–78.

[32] S. Böck, F. Krebs, and G. Widmer, "Joint Beat and downbeat tracking with recurrent neural networks," in *Proc. ISMIR*, 2016, pp. 255–261.

[33] J. Schulman et al., "Introducing ChatGPT," *OpenAI Blog*, 2022.

[34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[35] S. S. Stevens, J. Volkmann, and E. B. Newman, "A scale for the measurement of the psychological magnitude pitch," *J. Acoustical Soc. Amer.*, vol. 8, no. 3, pp. 185–190, 1937.

[36] G. Zhu, J.-P. Caceres, Z. Duan, and N. J. Bryan, "MusicHiFi: Fast high-fidelity stereo vocoding," 2024, *arXiv:2403.10493*.

[37] "DiffWave," 2023. [Online]. Available: https://github.com/lmnt-com/diffwave

[38] M. E. Davies, N. Degara, and M. D. Plumbley, "Evaluation methods for musical audio beat tracking algorithms," Queen Mary Univ., London, U.K., Tech. Rep. C4DM-TR-09-06, 2009.

[39] C. Raffel et al., "MIR _ EVAL: A transparent implementation of common MIR metrics," in *Proc. ISMIR*, 2014.

[40] Y. Wu, K. Chen, T. Zhang, Y. Hui, T. Berg-Kirkpatrick, and S. Dubnov, "Large-scale contrastive language-audio pretraining with feature fusion and keyword-to-caption augmentation," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2023, pp. 1–5.

[41] K. Chen, X. Du, B. Zhu, Z. Ma, T. Berg-Kirkpatrick, and S. Dubnov, "HTS-AT: A hierarchical token-semantic audio transformer for sound classification and detection," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2022, pp. 646–650.

[42] A. V. D Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," 2018, *arXiv:1807.03748*.
[43] K. Kilgour, M. Zuluaga, D. Roblek, and M. Sharifi, "Frechet audio distance: A metric for evaluating music enhancement algorithms," 2018, *arXiv:1812.08466*.
[44] S. Hershey et al., "CNN architectures for large-scale audio classification," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2017, pp. 131–135.
[45] J. F. Gemmeke et al., "Audio set: An ontology and human-labeled dataset for audio events," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2017, pp. 776–780.
[46] S. Forsgren and H. Martiros, "Riffusion: Stable diffusion for real-time music generation," 2022. [Online]. Available: https://riffusion.com/about
[47] K. Chen, Y. Wu, H. Liu, M. Nezhurina, T. Berg-Kirkpatrick, and S. Dubnov, "MusicLDM: Enhancing novelty in text-to-music generation using beat-synchronous mixup strategies," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2023, pp. 1206–1210.
[48] A.V.D. Oord, O. Vinyals, and K. Kavukcuoglu, "Neural discrete representation learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017.
[49] S. Dieleman, A. V. D. Oord, and K. Simonyan, "The challenge of realistic music generation: Modelling raw audio at scale," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018.
[50] L. Lin, G. Xia, J. Jiang, and Y. Zhang, "Content-based controls for music large language modeling," 2023, *arXiv:2310.17162*.
[51] Y.-K. Wu, C.-Y. Chiu, and Y.-H. Yang, "JukeDrummer: Conditional beat-aware audio-domain drum accompaniment generation via transformer VQ-VA," in *Proc. ISMIR*, 2022.
[52] H. F. Garcia, P. Seetharaman, R. Kumar, and B. Pardo, "VampNet: Music generation via masked acoustic token modeling," in *Proc. ISMIR*, 2023.
[53] K. Chen, C.-I. Wang, T. Berg-Kirkpatrick, and S. Dubnov, "Music SketchNet: Controllable music generation via factorized representations of pitch and rhythm," in *Proc. ISMIR*, 2020.
[54] H. H. Tan and D. Herremans, "Music FaderNets: Controllable music generation based on high-level features via low-level feature modelling," in *Proc. ISMIR*, 2020.
[55] S. Dai, Z. Jin, C. Gomes, and R. Dannenberg, "Controllable deep melody generation via hierarchical music structure representation," in *Proc. ISMIR*, 2021.
[56] S.-L. Wu and Y.-H. Yang, "MuseMorphose: Full-song and fine-grained piano music style transfer with one Transformer VAE," *IEEE/ACM IEEE Trans. Audio, Speech, Lang. Process.*, vol. 31, pp. 1953–1967, 2023.
[57] A.V. D. Oord et al., "WaveNet: A generative model for raw audio," 2016, *arXiv:1609.03499*.
[58] N. Kalchbrenner et al., "Efficient neural audio synthesis," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 2410–2419.
[59] C. Donahue, J. McAuley, and M. Puckette, "Adversarial audio synthesis," in *Proc. Int. Conf. Learn. Representations*, 2019.
[60] C. Hawthorne et al., "General-purpose, long-context autoregressive modeling with perceiver AR," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 8535–8558.
[61] Z. Borsos et al., "AudioLM: A language modeling approach to audio generation," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 31, pp. 2523–2533, 2023.
[62] D. Kim, Y. Kim, W. Kang, and I.-C. Moon, "Refining generative process with discriminator guidance in score-based diffusion models," in *Proc. Int. Conf. Mach. Learn.*, 2023.

Shih-Lun Wu received the B.Sc. degree in computer science from National Taiwan University, Taipei, Taiwan. He is currently working toward the research-track M.Sc. degree in language technologies institute, Carnegie Mellon University (CMU), Pittsburgh, PA, USA, where he is advised by Prof. Shinji Watanabe and Prof. Chris Donahue. During his degree, he spent a summer as a Research Scientist Intern with Adobe Research, where he was mentored by Dr. Nicholas J. Bryan. He was a machine learning research Engineer with Taiwan AI Labs, Taipei, Taiwan. His research interests include music and audio processing, and generative modeling. His research achievements have been recognized by NTU's university-wide Best Bachelor's Thesis Award, and the 2024 Siebel Scholarship.

Chris Donahue received the Ph.D. degree from the University of California San Diego, La Jolla, CA, USA, where he was jointly advised by Miller Puckette (music) and Julian McAuley (CS). He was a Postdoctoral Scholar with the Computer Science Department, Stanford University, advised by Percy Liang. He is currently an Assistant Professor with Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA, and a part-time research Scientist with Google DeepMind, London, U.K., working on the Magenta project. His research goal is to develop and responsibly deploy generative AI for music and creativity, thereby unlocking and augmenting human creative potential. In practice, this involves improving machine learning methods for controllable generative modeling of music and audio, and deploying real-world interactive systems that allow anyone to harness generative music AI to accomplish their creative goals through intuitive forms of control. His research has been featured in live performances by professional musicians such as The Flaming Lips, and also empowers hundreds of daily users to convert their favorite music into interactive content through his website Beat Sage. His work has also received coverage from MIT Tech Review, The Verge, Business Insider, and Pitchfork.

Shinji Watanabe received the B.S., M.S., and Ph.D. (Dr. Eng.) degrees from Waseda University, Tokyo, Japan. He is currently an Associate Professor with Carnegie Mellon University, Pittsburgh, PA, USA. He was a Research Scientist with NTT Communication Science Laboratories, Kyoto, Japan, from 2001 to 2011, a Visiting Scholar with the Georgia Institute of Technology, Atlanta, GA, USA, in 2009, and a Senior Principal Research Scientist with Mitsubishi Electric Research Laboratories, Cambridge, MA, USA from 2012 to 2017. Before Carnegie Mellon University, he was an Associate Research Professor with Johns Hopkins University, Baltimore, MD, USA, from 2017 to 2020. His research interests include automatic speech recognition, speech enhancement, spoken language understanding, and machine learning for speech and language processing. He is an Senior Area Editor of IEEE TRANSACTIONS ON AUDIO SPEECH AND LANGUAGE PROCESSING. He is an ISCA Fellow.

Nicholas J. Bryan (Member, IEEE) received the B.M. and B.S. degrees in electrical engineering (with summa cum laude, general, and departmental Hons.) from the University of Miami, Coral Gables, FL, USA, the M.S. degree in electrical engineering from Stanford University, Stanford, CA, USA, and the M.A. and Ph.D. degrees from CCRMA, Stanford University. He was a Senior Audio Algorithm Engineer with Apple. He is currently the Head of the Music AI Group and a Senior Research Scientist with Adobe Research, San Francisco, CA. He has authored or coauthored more than 34 peer reviewed papers, 13 patents. His research interests include music generation, machine learning, and signal processing. He was the General Co-Chair of WASPAA 2023, a two-time elected member of the IEEE Audio and Acoustic Signal Processing Technical Committee, and is an Adobe distinguished inventor. He was the recipient of two best paper awards, one AES Graduate Student Design Gold Award, and one best reviewer award.