Modeling Relations Between Musical Events in Continuous Time with Transformer Models for Live Co-Improvisational Interactions

Vincenzo Madaghiele
Department of Musicology
University of Oslo
vincenzo.madaghiele@imv.uio.no

Stefano Fasciani
Department of Musicology
University of Oslo
stefano.fasciani@imv.uio.no

Tejaswinee Kelkar
Department of Musicology
University of Oslo
tejaswinee.kelkar@imv.uio.no

Çağrı Erdem
Department of Informatics
University of Oslo
cagrie@ifi.uio.no

Abstract

This paper presents a live semi-autonomous system that co-improvises with a live musician using a model learned from the relationship observed in paired recordings of an improvising duo. This responsive system is based on a deep learning approach that models the relations between sequences of events produced by co-playing musicians in continuous time, using transformer models. We present the architecture for simultaneous sequences of sonic events and provide a quantitative evaluation on several representative tasks. Results are compared against a canonical transformer and a multichannel Factor Oracle, the latter being a widely used model for live sequence-based symbolic music generation. We detail a live implementation employing concatenative synthesis and introduce a customization procedure in which the generative model is iteratively retrained on curated, satisfactory sections from sound recordings of actual human-system co-improvisational interactions. This iterative fine-tuning enables the model's stylistic output to diverge from the original training corpus. Two musical use cases demonstrate the application of this technique.

1 INTRODUCTION

Sequence-to-sequence models have proven exceptionally effective at modeling natural language. However, a significant research gap exists in the use of these models for live, responsive generation of musical events. In music, sequence-to-sequence models have been primarily used for offline music generation tasks such as symbolic harmonization (Rhyu et al., 2022), counterpoint generation (Nichols et al., 2020), rhythm generation (Nuttall et al., 2021), and accompaniment (Agarwal and Sultanova, 2024). Online co-playing systems present additional challenges, such as tempo synchronization, calibration of feature extraction, generating simultaneous, low-latency responses, and higher adaptability of models to unexpected inputs. Moreover, models specialized in natural language can fail to capture the temporal dependencies unique to musical structures, such as the simultaneity of events on multiple parallel tracks or time densities, whose characteristics are highly dependent on the musical style.

Most proposed live co-playing systems address a scenario in which a human performer interacts with an artificial agent that listens to the performer's output and generates musical material in return. In these systems, response signals are generated either from a model of the input source (Tatar and Pasquier, 2017; Assayag et al., 2022), from a predefined temporal macro-structure (Scarlatos et al., 2025), or from a combination of both (Nika et al., 2017). While these approaches have been employed in various musical contexts with positive outcomes, co-playing systems developed along these lines exhibit several notable limitations (Nika et al., 2025). For example, such systems tend to be *reactive*, lacking independence, initiative, and a distinct style. Developing independence, context-awareness, and customizability in co-playing systems remains an open problem that recent works have begun to address using a variety of approaches (Scarlatos et al., 2025; Déguernel et al., 2018; Thelle and Pasquier, 2021; Bujard et al., 2025). However, these approaches frequently struggle to reconcile competing demands: the need for large training corpora—which limits customizability for artists wishing to train models on their own material—and model complexity, since architectures that can be trained on small datasets often fail to capture rich information such as long-term time dependencies.

In this paper, we propose a modified transformer architecture to model and generate musical events in continuous time. Our model departs from the canonical transformer by introducing learned *time2vec* encoding (Kazemi et al., 2020) to model temporal information, representing discrete events as vectors of continuous audio descriptors instead of token classes, and operating autoregressively with a time-based sliding window over recent events to capture local temporal context and enable real-time generation. In this way, we aim to address music's specific temporal dependencies and the variability across genres, musicians' styles, and instrumental timbres that determine which audio descriptors are salient. The use of descriptors introduces additional contextual information into the model, enabling better performance when training on smaller corpora compared with models that use raw audio encodings. Using continuous features extracted from audio as inputs has sev-

eral advantages with respect to symbolic representations: it simplifies data collection for ordinary musicians, and it allows for retaining timbral nuances, focusing on aspects of music that might not be captured by symbolic relations, increasing flexibility across musical genres.

We developed the model to be trainable from scratch on relatively small corpora collected by musicians, and our choices of data representation and architecture reflect this constraint. The approach is genre- and instrument-agnostic: it does not presuppose any time quantization, and it represents sound using audio descriptors, making it suitable for both conventional and experimental applications. In this paper, we detail the model architecture and the corresponding data representation, provide a quantitative comparison with a canonical transformer and a multichannel Factor Oracle (FO) model, and introduce a fine-tuning procedure based on iterative interaction and selection of satisfactory outputs. Finally, we demonstrate the technique in two musical scenarios.

2 BACKGROUND

2.1 Modeling musical events in continuous time

Timing is a fundamental aspect of making music together. As performers play, music flows in time as a continuous sequence of events. Constructing mathematical models of this flow of events has been a popular research endeavor since the beginning of computer music. Modeling musical events in time enables the prediction of the musical action that will follow a given action by a specific musician in a particular context (Dubnov et al., 2003). Obtaining a perfect model of a musician's decision-making is inherently challenging, since a musician's style is complex, dynamic, and influenced by non-musical factors and human experiences. However, despite their imperfections, such models can be used to generate musical events during performances and have been applied in avant-garde music compositions (Xenakis, 1971), live electronics (Panariello and Déguernel, 2026), and computer music (Ghisi and Agostini, 2017). In recent years, these models have begun to become accessible to a broader community of musicians (Roberts et al., 2019).

Several approaches have been employed to develop mathematical models of music improvisation, the most popular being information-theoretical (Dubnov, 2021), Markovian models, and deep neural networks. Among the latter, sequence-to-sequence (seq2seq) models have recently gained significant popularity due to their outstanding performance on natural language.

Sequence-to-sequence (seq2seq) models are machine-learning architectures that learn mappings between input and output sequences by encoding the source sequence into a latent representation and decoding that representation to generate the target sequence, thereby capturing temporal and contextual relations between events. Among seq2seq models, transformers have achieved exceptional results in recent years when applied to natural language (Vaswani et al., 2017), video (Selva et al., 2023), and speech (Karita et al., 2019). Despite these advances, the application of transformers to time series regression remains underexplored, possibly because the canonical transformer is not explicitly designed to model continuous, time-dependent features. State-of-the-art music transformers commonly utilize rotary positional embeddings (RoPE) to capture temporal dependencies (Su et al., 2024);; however, these encodings do not capture continuous temporal information such as event durations, inter-event intervals, or temporal relations among simultaneous events. These temporal attributes are fundamental to music (Roads, 2003; Clarke, 1987). Prior works have addressed them by introducing separate time embedding for events duration (Madaghiele et al., 2021; Hakimi et al., 2020); however, this strategy requires quantizing durations into discrete classes, which may be unsuitable for certain musical genres. Time2vec (Kazemi et al., 2020) offers an alternative approach for representing time in deep neural networks by learning time representations— including periodic components—directly from data. It has shown improved prediction performances on continuous time series and irregularly sampled events (Diniz et al., 2022; Bispo Junior et al., 2025; Xiao et al., 2025) but, to our knowledge, it has not yet been applied to musical data.

2.2 Responsive models in co-improvisation systems

Artificial co-improvisation systems are interactive music systems (Rowe, 1999) able to listen to a live improvising musician, process the sound produced by the musician and compute appropriate responses by controlling a sound-making model (Blackwell et al., 2012). Co-improvisation systems are strongly performance-driven (Drummond, 2009), meaning that there is no (or a limited amount of) macro-temporal structure determining the course of the performance, and therefore they need to be *causal*, i.e., model responses are only dependent on the past actions of the musician. The co-creative scenario (Veale and Cardoso, 2019; Carnovalini and Rodà, 2020) in this setting usually prescribes a live improvising musician and a computer program listening and generating music, which can be general-purpose, such as the Voyager (Steinbeck, 2018), Somax (Assayag et al., 2022), Dicy (Nika et al., 2017) and MASOM families of systems (Tatar and Pasquier, 2017), or specialized to tasks such as harmony (Scarlatos et al., 2025) and rhythm generation (Vigliensoni et al., 2022; Haki et al., 2022).

Techniques for generating musical responses range from rule-based systems to machine-learning models, including Markovian approaches such as Factor Oracle (FO) (Assayag and Dubnov, 2004) and Variable Markov Oracle (VMO) (Wang and Dubnov, 2014; Wang et al., 2016), as well as deep-learning architectures such as recurrent neural networks (RNN) (Minu et al., 2022), long-short term memory (LSTM) (Erdem et al., 2022), variational autoencoders (VAE) (Dubnov, 2021), and transformers (Scarlatos et al., 2025; Haki et al., 2022). These models are frequently employed to generate music in the symbolic domain (e.g., MIDI), or to predict classes of sounds obtained by clustering a predefined corpus of sounds according to similarity over a set of audio descriptors (Tatar and Pasquier, 2017; Assayag et al., 2022). These predictions determine the choice of synthesis engine: symbolic (MIDI) predictions permit the use of MIDI-compatible synthesizers, whereas class-based sound predictions are typically realized via concatenative synthesis, in which exemplar samples from the predicted classes are selected and concatenated to produce sound.

Current approaches to modeling responses in co-improvisation typically presuppose a dyadic improvisational setting, but in practice, most implement only a sequence model trained on the musician's behavior, which the agent then uses to generate responses. This approach

does not consider that the style of the input musician and that of the artificial agent can be distinct and relational, significantly limiting the range of possible responses, since the artificial agent is forced to be *reactive* (Nika et al., 2017; Scarlatos et al., 2025). This issue had been recognized and addressed in recent works by augmenting baseline architectures with explicit control mechanisms (Thelle and Pasquier, 2021), by enabling inter-model communication via message-passing among parallel models (Déguernel et al., 2018), or by constructing an autonomous model that operates independently while being modulated by the human performer's actions (Bujard et al., 2025; Pasini et al., 2026; Jiang et al., 2020; Benetatos et al., 2020). Here we address this issue by training a seq2seq model on paired co-playing audio tracks—a source (the human) and a target (the agent)—so the agent learns to generate events in the style of the target while being influenced by the source. After training, the agent's behaviors can be fine-tuned via iterative interactions and selective feedback. Unlike prior approaches that model only a single performer or impose control scaffolding on an otherwise reactive module, our model directly learns how one performer's sequence of events affects another's, capturing inter-track relations while preserving rich, multi-dimensional contextual information. This enables the agent to imitate both a target style and respond adaptively to the source in a way that supports more independent and stylistically consistent co-improvisation. Modeling relationships between musical events on multiple tracks is essential for developing more complex behaviors beyond imitation in artificial musical agents (Dahlstedt, 2021; Ben-Tal and Dolan, 2023; Gifford et al., 2018). While our initial model is trained on imitating the multi-track relations found in a target corpus, we enable divergent behaviors by proposing an adaptation method based on iterative co-playing and fine-tuning, allowing the model to personalize beyond the original training data.

3 DATA REPRESENTATION

The proposed model is trained on paired audio tracks of co-playing musicians. Such pairs can be acquired in several ways: by multi-track recording a duo performance, by overdubbing a new part while playing along with an existing track, or by rendering pre-composed MIDI parts into audio with virtual instruments. A training corpus may contain many such track pairs; crucially, each pair must be time-aligned so that the model can learn the temporal relationships between the two performers. We assume that the two musical tracks are produced by separate co-playing musicians and, for simplicity, that each musician produces a monophonic stream of events. In other words, the model does not represent polyphony of a single track: when multiple sounds occur simultaneously (e.g. a strummed guitar chord, or two drum-kit elements struck together, such as kick and hi-hat), those acoustic occurrences are represented as a single event with an audio descriptor that aggregates the simultaneous sources, since descriptors are computed on the mixed signal containing both events. This design choice simplifies the event representation and training, while still allowing complex sonic events to be encoded via multidimensional descriptors.

We denote the pair of audio tracks by $x(t)$ and $y(t)$. In the descriptions that follow, $x(t)$ refers to the human performance (source) and $y(t)$ to the model output (target). For simplicity of exposition we present the signals in continuous time, although the implemented system operates in discrete time. After collection, we process each track independently to extract onset times and events, defined as segments of audio signals between two consecutive onsets. We use a low-latency onset detector inspired by the Data Knot library for MaxMSP and Pure Data (Costanzo, 2025). Let N and M be the number of events in $x(t)$ and $y(t)$, respectively. We obtain ordered onset sets for each track

$$\begin{aligned} \mathrm{ONSETS}\big(x(t)\big) &= \{\, o_{x,n} \mid n = 0, \ldots, N-1 \,\}, \\ \mathrm{ONSETS}\big(y(t)\big) &= \{\, o_{y,m} \mid m = 0, \ldots, M-1 \,\}. \end{aligned} \tag{1}$$

Each $o_{x,n}$ and $o_{y,m}$ is an onset time, and indices n and m increase with time. A sound event is therefore the segment of audio between two consecutive onsets. For $x(t)$ and $y(t)$, events are defined as

$$\begin{aligned} e_{x,n}(t) &= x(t) \quad \text{for} \quad o_{x,n} \le t < o_{x,n+1}, \\ e_{y,m}(t) &= y(t) \quad \text{for} \quad o_{y,m} \le t < o_{y,m+1}. \end{aligned} \tag{2}$$

Event durations are therefore

$$\begin{aligned} d_{x,n} &= o_{x,n+1} - o_{x,n}, \\ d_{y,m} &= o_{y,m+1} - o_{y,m}. \end{aligned} \tag{3}$$

We compute the following audio descriptors for each event $e_{x,n}(t)$ and $e_{y,m}(t)$: root-mean-square (RMS) amplitude, spectral centroid, spectral flatness, spectral rolloff, pitch, chroma, and 13 Mel-Frequency Cepstral Coefficients (MFCC) coefficients. The descriptors are computed on frames of 1024 samples with 50% overlap, at sample rate of 44.1 kHz. Following a Bag-of-Frames (BoF) approach (Aucouturier et al., 2007), we represent each event by the mean and standard deviation of each descriptor taken over the frames inside the event. The resulting feature vector for the n-th event $e_{x,n}$ is denoted $\mathbf{v}(e_{x,n})$ and includes these aggregated descriptor statistics together with the event duration, $d_{x,n} = o_{x,n+1} - o_{x,n}$. RMS amplitude is also used to split an event into subsegments when the RMS falls below a threshold of 2% relative to the event's peak, indicating a segment of silence prior to the next onset. For such silent subsegments, we set all audio descriptors to zero and assign their start time to the moment when silence is detected. We impose minimum and maximum event durations of 20 ms and 2 s, respectively, to avoid excessively long or short events. These thresholds are flexible and can be tuned to the musical style of the corpus. Events shorter than the minimum are merged with the preceding event, while longer than the maximum are split into two consecutive events (recursively if necessary until all segments fall within the allowed range). The result of preprocessing for each event is feature vector $\mathbf{v}(e_{x,n})$ defined as follows:

$$\begin{aligned} \mathbf{v}(e_{x,n}) &= \{\mathbf{D}(e_{x,n}), d_{x,n}\}, \\ \mathbf{v}(e_{y,m}) &= \{\mathbf{D}(e_{y,m}), d_{y,m}\} \end{aligned} \tag{4}$$

where $\mathbf{D}(e_{x,n})$ are the BoF audio descriptors computed from the event, and $d_{x,n}$ is the event's duration.

A subset of the computed descriptors is used for training; the particular subset is chosen according to the characteristics of the sound material. Different descriptor subsets may be used for the source and the target signals to better

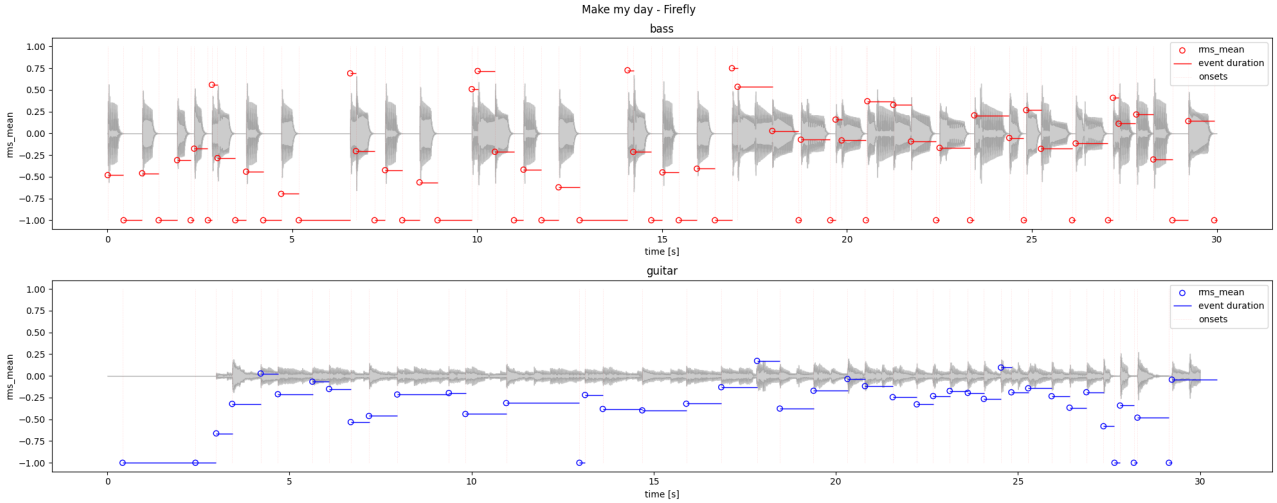

Figure 1: Data representation in our model. The tracks in the figure are the bass and guitar stems of the song *Make My Day*, by artist *Firefly* from MoisesDB (Pereira et al., 2023). Musical information of each separate audio track is represented as a monophonic sequence of sonic events. An event corresponds to an onset detected in a track, or silence when the event's loudness gets lower than a threshold; each event is encoded as a set of continuous sound descriptors, and its duration. In this figure, the y-axis shows the mean RMS amplitude of each event (one of the audio descriptors), normalized to the range $[-1, 1]$ for display. The line length corresponds to the event's duration, while the onsets are indicated by the left end of each horizontal line.

match their timbral/expressive roles. An illustration of the data encoding is shown in Figure 1.

We represent sound as discrete events encoded by handcrafted audio descriptors because this approach does not require large datasets or pre-trained embeddings (e.g., wav2vec (Ragano et al., 2023)) to learn useful representations. By extracting compact, interpretable audio descriptors per event, we reduce data and compute requirements while retaining musically relevant information for modeling and interactive use. Our decision to use vectors of continuous quantities instead of discrete embeddings broadens the model's applicability: it can directly control different sound-generation systems (MIDI and/or concatenative synthesis) while still being able to produce event classes when required, as demonstrated in Section 5.3.2. Providing the model with direct access to descriptor values—rather than forcing it to generate class labels derived from those descriptors—enables it to learn richer, more nuanced relationships between events.

4 MODEL ARCHITECTURE

We propose a deep-learning architecture to model the relations between two simultaneous streams of events in continuous time, to predict events in one stream conditioned on the other. Each stream corresponds to a co-improvised musical part, presented to the model as parallel encodings. Our model adapts the canonical encoder-decoder transformer of (Vaswani et al., 2017) by using Time2vec (Kazemi et al., 2020) for time encoding instead of the standard positional encoding, and replacing token embedding layers with linear projection layers so the network encodes and predicts vectors of continuous descriptors rather than discrete token classes, i.e., it performs a regression. The model is trained using a Mean-Squared Error (MSE) loss for backpropagation.

The model is trained as a regression task, aiming to predict, for each event in the target sequence, the continuous values of the features of the next target event. The inputs of the model are the latest subsequences of source and target events $\overline{X}_m$ and $\overline{Y}_m$, identified by a fixed-time sliding window of duration T. For each new event to be predicted $\hat{\mathbf{v}}(e_{y,m+1})$, the model uses as inputs the events in the T seconds of audio preceding the start of the new target event. Considering a latest target event $e_{y,m}$, the model predicts $\hat{\mathbf{v}}(e_{y,m+1})$ using the following subsequences as inputs:

$$\begin{aligned}\overline{X}_m &= \{\mathbf{v}(e_{x,i}) | i \quad \text{s.t.} \quad o_{y,m+1} - T \le o_{x,i} < o_{y,m+1}\},\\ \overline{Y}_m &= \{\mathbf{v}(e_{y,j}) | j \quad \text{s.t.} \quad o_{y,m+1} - T \le o_{y,j} < o_{y,m+1}\}.\end{aligned} \tag{5}$$

where i and j are two independent discrete indices denoting the events included in the time window preceding to $o_{y,m+1}$.

Here $o_{y,m+1}$ is known at inference time, being equal to the latest onset time $o_{y,m}$ plus the duration of the latest event in the target sequence $\hat{d}_{y,m}$. At each inference step, the time window moves forward by the predicted duration of the previous event. This makes the system causal and autoregressive during inference: it generates new events based on the previously generated past events and on the source events produced by the live musician in the meantime.

4.1 Time2vec encoding

Time2vec (Kazemi et al., 2020) is a time encoding that learns a vector representation of time that can model periodicities and is invariant to time rescaling. Time2vec has been used as a trainable time layer in several time-series models including LSTMs (Peñaloza, 2020; Geng et al., 2023) and transformers (Diniz et al., 2022). Given a scalar time feature τ, the time2vec embedding $\mathbf{t2v}(\tau) \in \mathbb{R}^{k+1}$ is a vector of $k+1$ learnable components defined as follows:

$$\mathbf{t2v}(\tau)[z] = \begin{cases} \omega_0\,\tau + \varphi_0, & z = 0, \\ \mathcal{F}\big(\omega_z\,\tau + \varphi_z\big), & z = 1, \ldots, k \end{cases} \tag{6}$$

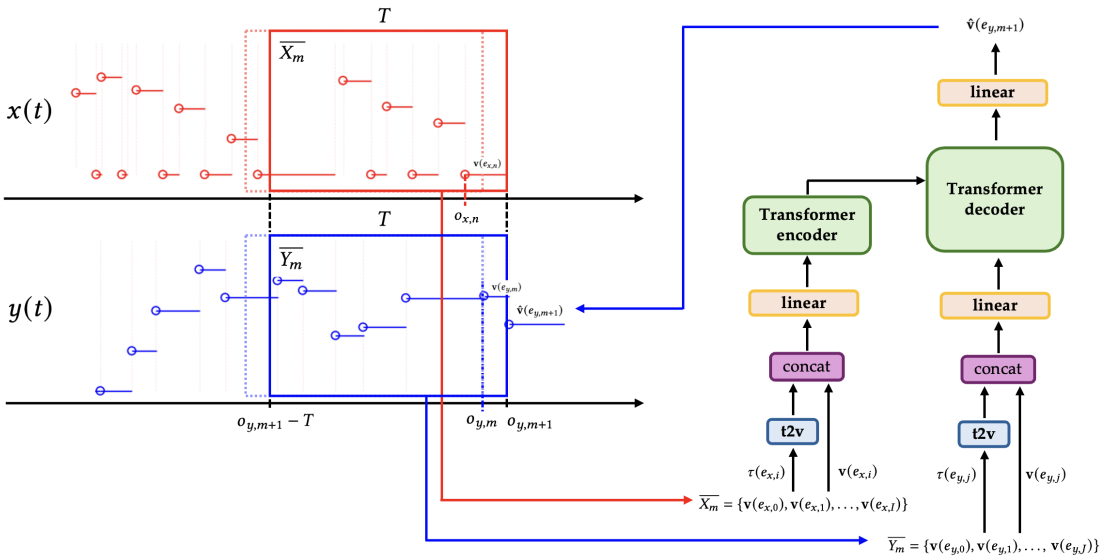

Figure 2: Block diagram of the deep-learning model architecture. Events in source and target subsequences $\overline{X}_m$ and $\overline{Y}_m$ are selected from a time interval of duration T to predict $\hat{\mathbf{v}}(e_{y,m+1})$, the descriptor vector of the next target event. Each subsequence is associated with a τ feature vector representing local time of each event. After time2vec encoding, the time-augmented sequences are processed by a regression transformer that outputs the descriptors and the duration of the next target event.

where $\mathcal{F}$ is a periodic activation function. In our case, $\mathcal{F}(\cdot) = \sin(\cdot)$, and ω_i, φ_i are learnable parameters controlling the frequency and phase of each component. The embedding therefore has one affine component ($z = 0$) and k periodic components; the total dimension is $k + 1$, with k chosen according to the application.

In our model, τ features are recomputed for each prediction to encode the time distance from every event in the input window to the start of the event being predicted. Let $o_{y,m+1}$ denote the onset time of the next target event to be predicted. Then for each source event $e_{x,i}$ and each past target event $e_{y,j}$ in the window we define

$$\begin{aligned} \tau(e_{x,i}) &= o_{y,m+1} - o_{x,i}, \\ \tau(e_{y,j}) &= o_{y,m+1} - o_{y,j}. \end{aligned} \tag{7}$$

so that τ measures how far in time each event is from the preedicted event start. The τ feature is computed separately for source and target events, using a moving window of duration T; the window is positioned so that its end aligns with the start time $o_{y,m+1}$ of the next target token, as shown in Figure 2.

4.2 Regression output

For each event the computed τ feature is concatenated to the vector of audio descriptors. Source and target subsequences may contain different numbers of events, up to a maximum sequence length L chosen based on the training data. Sequences shorter than L are padded with a pad token at the end of the sequence; these pad positions are ignored during training via pad masking.

As illustrated in Figure 2, separate time2vec encoders are learned for source and target subsequences. After time2vec encoding, the resulting $\mathbf{t2v}$ features are concatenated directly with the feature vector $\mathbf{v}(e_{x,n})$ and passed through a linear projection layer. The projected source sequence is processed by a transformer encoder whose outputs condition a transformer decoder that models the target sequence. The decoder regressively predicts the continuous features of the next event, including the event descriptors and their duration.

During training we optimize the MSE between predicted ground-truth descriptor vectors. At inference (live co-improvisation) the predicted event is synthesized/played and appended to the target event sequence so it can be used as input for the next autoregressive prediction.

5 EXPERIMENTS

In this section, we present a quantitative evaluation of the proposed model. The architecture was designed for flexibility and for training with limited data, so we assess its performance across several diverse, relatively small corpora.

5.1 Datasets

We evaluate the model on a pair of stems extracted from the MoisesDB dataset (Pereira et al., 2023), which was originally compiled for source-separation tasks. To the best of our knowledge, there is no openly accessible dataset of improvising duos large enough to conduct a quantitative evaluation across multiple sound sources. Consequently, we chose MoisesDB because it contains parallel recordings of several instruments, allowing us to test the flexibility of the model across different pairings of sonic materials. The audio recordings in MoisesDB are not improvised, and learning the relationship between two tracks while ignoring the remaining stems might affect the model performance; these factors should be considered when interpreting the quantitative results. To obtain training corpora with stylistic homogeneity, we selected a subset consisting of nine tracks by the artist *Firefly*, whose genre is all labeled as *singer-songwriter*. For each track, we used the available stems for *guitar*, *bass*, and *drums*, yielding a total duration

of approximately 30 minutes. A window size of 1024 samples for analysis is not sufficient for appropriate detection of pitch and chroma in the lower register of the bass, however in the bass tracks chosen for evaluation are mainly in the medium register, not affecting the results. It is indeed possible for the system to work with a larger window size, if it is needed to detect bass notes in the lower register. When a drum recording contained multiple stems, those stems were mixed (i.e., summed and divided by their count) to produce a single mono drum signal for the track. The specific track names are listed in the supplementary material to ensure reproducibility. All experiments reported in this section are performed on all ordered pair permutations of these stems drawn from the selected tracks; we treat these pairs as stylistically homogeneous couples for training and evaluation.

5.2 Training

We train our model on all combinations of the selected stems. The model is implemented entirely in Python, using the Pytorch library for machine-learning, and the Librosa library (McFee et al., 2015) for extraction of audio descriptors. The sequences of events extracted from the training data are processed by computing the subsequences $\overline{X}_m$ and $\overline{Y}_m$ corresponding to each ground-truth target event $\mathbf{v}(e_{y,m+1})$. We establish the maximum length L of subsequences as the mean subsequence length plus their standard deviation. Shorter subsequences are padded, and longer subsequences are truncated, keeping only the events closest to $e_{y,m+1}$. The same maximum sequence length is used for source and target subsequences, and it is the input sequence length of the model. Subsequences are then randomly shuffled. The data is split into 80% training and 20% test, keeping 10% of the training data for validation. Models are trained with mean-square error (MSE) loss function for 3000 epochs, using a batch size of 16, employing stochastic gradient descent with a learning rate of 0.001 and dropout coefficient of 0.3. All models compared here share the same hyperparameters, which were chosen via a preliminary grid search; different hyperparameter settings may be preferable for other musical styles or instruments.

The hyperparameters used for training are the following: $k = 8$ time2vec frequency components, 32 neurons in the first linear layers for encoder and decoder, 8 parallel encoder and decoder heads for multi-head attention, 8 encoder and decoder layers, 512 neurons in the feedforward layers of encoder and decoder, and a window size $T = 8$ seconds. Moreover, different subsets of audio descriptors were selected based on the specific instruments and heuristics considerations; the descriptors used for each pair are listed in Table 1. The models were trained on a workstation featuring an Intel Core i9-14900KF CPU with 24 cores, an NVIDIA GeForce RTX 4090 GPU with 24 GB of memory, and 192 GB DDR5-5200 RAM, running Ubuntu 24. Training takes approximately 7 hours on this workstation; the same training requires roughly 12 hours on a MacBook Pro with an Apple M2 Pro chip with 16 GB of memory, making the method accessible to musicians without exceptional computational resources.

5.3 Performance

5.3.1 Model convergence over different instruments

In Table 2, we report the comparative results over different corpora extracted from MoisesDB. For each combination, we report the MSE between each predicted feature vector and the ground truth, and the multi-resolution STFT (mrSTFT): a sound-based metric of the difference between ground truth and the sample extracted for concatenative synthesis based on the prediction (Mateo and Talavera, 2020).

Table 1: Combination of audio descriptors selected for each instrument.

Instrument	Sound descriptors
bass	RMS, centroid, flatness, rolloff, pitch, chroma
guitar	RMS, pitch, chroma
drums	RMS, centroid, flatness, rolloff

As shown in Table 2, results are fairly consistent across corpora with different instruments. We observe a significantly higher value of mrSTFT when the model learns to predict drums. The reason for this shortcoming could be due to the fact that predicting a wrong drum component (e.g., a cymbal instead of a snare) generates widely different mrSTFT with respect to predicting wrong sounds of bass and guitar, whose timbre is more homogeneous. Based on a listening analysis, the model seems to capture short-term dependencies quite well, while it still struggles to model longer-term repetitive structures like rhythms. This is probably because of the length of the context time window $T = 8$ seconds. This time is enough to include short-term variations in the context, but can't include meso-structural repetitions like rhythms and chord progressions.

These findings are specific to the singer-songwriter genre we selected, and performances may vary on different genres. This genre often has macro-structural cues (e.g., repeated chord progressions and sections) and strong pulsed rhythmic contents. While time2vec can, in principle, capture some of these temporal regularities, better performance on idiomatic genres might be obtained by adding features that explicitly encode macro-structure, such as chord progression and position within a beat. Finally, although we treat data as duo performances, the original tracks contain other instruments whose contributions the model ignores; this lack of context may also degrade prediction performance.

5.3.2 Comparison with FO and canonical transformer

Factor Oracle (FO) is a sequence model originally developed for string matching in genetics and biology (Allauzen et al., 1999), later adapted to modeling musical sequences (Assayag and Dubnov, 2004). FO and related approaches are popular in live co-improvisation systems because of their strong performances on small corpora and their fast training and inference times. However, these models have a few shortcomings: they don't retain long-term memory dependencies, and they operate only in the symbolic domain. In this quantitative evaluation, we compare the performances of our model with respect to a multichannel FO implemented as in (Assayag and Dubnov, 2004) and against a canonical transformer.

The model we propose predicts continuous feature values; however multichannel FO and transformers predict symbolic tokens. To compare them, we discretize the continuous feature space into $C = 30$ classes using the Birch clustering algorithm (Zhang et al., 1996). Similar

Table 2: Results of the training process over different combinations of instrumental stems extracted from the same subset of MoisesDB. For each combination, we report MSE, multi-resolution STFT (mrSTFT) and coefficient of determination (R^2) measured on test tracks from the corresponding subset that were not used for training.

Source	Target	MSE		mrSTFT		R^2	
		validation	test	validation	test	validation	test
bass	drums	0.508	0.572	$8.030 \cdot 10^3$	$8.573 \cdot 10^3$	0.342	0.176
drums	bass	0.745	0.693	8.667	8.688	0.198	0.126
bass	guitar	0.558	0.626	9.924	9.891	0.089	0.085
guitar	bass	0.648	0.607	9.288	9.317	-0.855	-0.845
drums	guitar	0.524	0.648	10.133	9.689	0.185	0.014
guitar	drums	0.501	0.569	$7.079 \cdot 10^3$	$7.699 \cdot 10^3$	0.315	0.200

clustering-based classification approaches are used in (Assayag et al., 2022; Nika et al., 2017; Tatar and Pasquier, 2017). In Table 3, we report the prediction accuracy and mrSTFT for each dataset and model. The mrSTFT is computed by selecting a random sample from the predicted class in the training corpus, as it would be done in a live scenario.

For the multichannel FO implementation, we construct a Factor Oracle whose states correspond to tokens in the target sequence. For the source, we encode T seconds preceding each event by averaging the descriptor values of the event in that window, and we add a feature that counts the number of events (whose value is normalized before clustering, together with other features); these source vectors are clustered separately from the target sequence. This yields a source encoding sequence that is aligned with the target states, it has the same number of events as the source, and it is usable in a live scenario.

Because FO builds a separate model for each training sequence, for each test sequence we run prediction with all FOs built from the training set and select the token predicted by the plurality of models (an ensemble voting scheme, similar to bagging (Odegua, 2019)).

The accuracy values are in line with those reported in previous works on similar symbolic music-generation tasks (Bujard et al., 2025; Hakimi et al., 2020; Madaghiele et al., 2021). Although the accuracy scores are relatively low, it is important to note the limitations of accuracy for this task: it does not account for incorrect class predictions that are nevertheless closer to the ground truth than other classes. Overall, FO performs worse than the transformer-based models, likely because FO does not generalize across multiple training subsequences.

The table shows how our model's performance compares to the two chosen baselines. As shown, our model achieves substantially better mrSTFT score, indicating a substantial improvement of the regression-based approach with respect to clustering-based methods, in terms of accuracy, generating events that are closer to the ground truth than those produced by FO and the canonical transformer. Accuracy scores of the canonical transformer and the model we propose are comparable, with our model being significantly better at predicting drums, and slightly worse on other instruments. Note that both accuracy and mrSTFT are strongly influenced by the number of classes: increasing the number of classes can yield more precise (and thus better-scoring) mrSTFT samples, while potentially reducing classification accuracy for FO and the canonical transformer. These results do not mean that FOs should not be used: the decision between FO and transformer should probably depend on the availability of data, as transformers' performances generally scale well as more data is available, while FOs can achieve good levels of prediction with a low amount of data. These results do not demonstrate the superiority of our model in a classification task; however, they show that our model reliably enables regression-based inference of sound descriptors and continuous event durations. This introduces novel possibilities for descriptor-based interactive control of live synthesis, as well as continuous event timing untied from fixed duration and metric grids.

6 LIVE IMPLEMENTATION

The live implementation of the proposed system is written entirely in Python. We use PyTorch for model inference and training, Librosa (McFee et al., 2015) and custom code for computation of audio descriptors, and the Signalflow library (Jones, 2025) for live sound processing. In a live setting, the incoming source signal is continuously analyzed in short windows of 256 samples with 50% overlap. All other parameters regarding which sound descriptors to use, normalization, maximum sequence length, time window size T, sample rate, FFT window size, and hop size are the same as those used for training. At the start of a performance, both source and target sequences are initialized with silence events. The live program continuously detects onsets in the source stream, determines when an event has ended, and computes the event descriptors. When an event ends in the source track, its descriptor vector and duration are appended to the source sequence of source events. Each time the model generates a new target event, the events outside the latest T seconds are removed from the source and target sequences, as they are not used for inference. An event is considered to have ended either when its RMS amplitude falls below 2% relative to the peak value in the same event, or when a new onset is detected; in the former case, a silence event is inserted until the next detected onset. The same preprocessing used for data preparation is applied to event subsequences before inference: the same minimum and maximum event durations are used, the same features are computed, T window size, maximum sequence length, and the events are normalized using the same coefficients applied for training.

The model generates events autoregressively, conditioning on the source events and predicted target events that occurred in the most recent T seconds. After the model generates an event, that event's descriptor vector and duration are appended to the target sequence, and the model immediately generates the next event using source and target subsequences that span the previous T seconds. To avoid cumulative latency, a new prediction is produced 10ms

Table 3: Comparative performances of our model with respect to multichannel FO and canonical transformer. All reported results correspond to test data comprising to 20% of the total corpus. We report accuracy, mrSTFT and F1 score for each corpus and model.

Source	Target	FO			Transformer			Proposed		
		accuracy	mrSTFT	F1	accuracy	mrSTFT	F1	accuracy	mrSTFT	F1
bass	drums	3.169%	192.813	0.041	28.891%	31.559	0.264	41.450%	$8.573 \cdot 10^3$	0.395
drums	bass	4.788%	294.601	0.011	25.304%	33.132	0.213	23.932%	8.688	0.208
bass	guitar	5.874%	384.079	0.011	23.529%	79.368	0.187	17.892%	9.891	0.153
guitar	bass	10.424%	51.490	0.064	29.344%	51.202	0.241	22.561%	9.317	0.182
drums	guitar	7.071%	91.005	0.023	23.406%	29.698	0.188	17.156%	9.689	0.145
guitar	drums	2.150%	334.287	0.013	28.183%	30.068	0.255	42.511%	$7.699 \cdot 10^3$	0.418

before the end of the current target event—10ms being the worst-case scenario model's inference time. Therefore, during training, we subtract 10ms inference time from the duration of the last event so that predicted timings match the live inference schedule. On a modern laptop, inference time is around 3ms using CPU, however we chose 10ms such that the same results might apply to much less powerful laptops. Nevertheless, the effect of this parameter on the training performances is negligible. When the model generates a new event, the corresponding sound is produced using concatenative synthesis: the model predicts the event's descriptor vector, and we select and play the training corpus sample $e_{y,m}$ whose sound descriptors most closely match the predicted vector. We use Euclidean distance as a distance metric to find the closest match. This approach can cause timing issues because the duration of the chosen sample may not be precisely equal the duration predicted by the model, which can disrupt strict rhythmic alignment. Possible remedies include using a corpus with quantized (or duration-matched) samples, time-stretching/cutting the selected sample to the predicted duration, or adding a post-processing stage to enforce timing constraints; these extensions are not implemented here[1].

7 CASE STUDIES WITH FINE-TUNING

After training, a transformer model's behavior can be adapted to a particular style or task by iteratively fine-tuning on a specialized dataset. Fine-tuning continues training a model that was pretrained on general data, so it better matches a smaller target dataset. Our goal is to steer the model's behavior towards the aesthetic preferences of a musician by using the musician's own live interactions with the system as fine-tuning data. We therefore propose the following workflow. First, the musician trains a baseline model on pre-recorded data. Next, they co-perform with the live model and record paired musician and model tracks from several sessions. The musician then reviews these recordings and trims them to retain only the segments where the model's behavior is judged satisfactory according to the musician's own preferences. These selected segments form a focused corpus for continued training of the baseline model, producing a new model more closely aligned with the musician's preferences. In the rest of this section, we present two case studies that illustrate this process.

[1]The code for the model and our live system are released as open-source software and are available at the paper's companion page, together with audio examples of the generated tracks and video examples https://github.com/vincenzomadaghiele/co-impro-transformer.

7.1 Case study I: Guitar duo

This case study models a duo performance with two guitars. The data were collected by the first author by recording a solo guitar track in one take and then overdubbing it with another improvised take, obtaining paired tracks as if they were from a duo. The corpus consists of five duo performances of duration around 2 minutes each, for a total duration of around 10 minutes. The aesthetic interest of this experiment lies in attempting to capture the intuitive choices made by a musician who improvises over a track they recorded moments earlier. When improvising the second track, the musician retains an embodied memory of the previous improvised track and is in the same overall mood, creating a feeling of intuitive convergence between the two tracks. Tracks in the corpus have a free tempo and include *crescendo* and *diminuendo* sections, and they all revolve around the same tonal center (D Ionian mode) with episodic excursions in different modes and dissonances. The objective of this case study is to obtain a model able to replicate global relations between tracks, for example, when the musician plays on the high register, the agent should play on the lower one, and when the tempo is speeding up, the agent should react accordingly, found in the fine-tuning data.

The model was trained using the first recorded track as a source and the second recorded track as a target. Input descriptors comprised RMS, spectral centroid, spectral flatness and chroma to capture dynamics, timbral characteristics, instrumental register, and harmonic relations. After the initial training, the model's behavior was not yet coherent enough for musical interaction. Two strategies were then possible: collect more data, or make the corpus more internally coherent by removing parts that are stylistically different. In this case, we applied both strategies and retrained the model until we obtained a satisfactory baseline for fine-tuning. This is important: to be able to fine-tune a model interactively, its behavior must already be at least acceptable to play with. After playing with the model, we selected sections where the roles of melody and accompaniment relative to the source were clearly defined and fine-tuned the model based on those sections for 1000 epochs. This process was repeated two times, until we considered the result satisfactory.

7.2 Case study II: TapBot

TapBot is a performance for finger tapping between a solo musician and an artificial model. The musician taps on a wooden block, whose sound is recorded with a contact microphone. The agent responds by controlling a linear push/pull solenoid that produces a tap on the second identical wooden block. A prototype of the system is portrayed

in Figure 3. The aesthetic objective is again to capture the embodied intuition of a performer. In this case, the performer records the corpus by finger-tapping on two wooden blocks simultaneously with different hands, recording the two signals on two separate tracks that are then used as source and target during training. The agent is trained using the right-hand taps as the source and predicting the left-hand taps as the target.

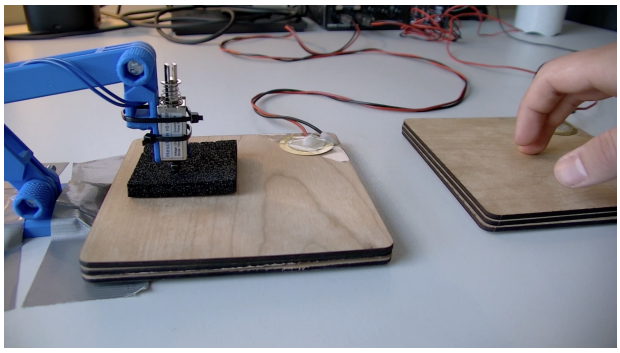

Figure 3: Prototype of the hardware system used for the TapBot performance. A solenoid, visible on the left, is activated by sound impulses sent from the Python software, corresponding to tapping events generated as the target signal. The system listens to the musician's taps on the wooden board visible on the right, using those events as the source signal.

The model was trained using only RMS, so in practice, it aims to predict dynamics and duration of rests between two tapping events. During performance, the model's predictions are used to control the timing and intensity of the bot's taps on the wooden block. Fine-tuning focused on selecting interactive, rhythmical, and varied call-and-response behaviors, while discarding aesthetically unclear interactions. Because this performance is strongly rhythm-based, the model's weakness in capturing long, repeated rhythmical patterns is limiting: the model does not reliably lock into a prolonged rhythm.

8 CONCLUSIONS

We have introduced a sequence-to-sequence deep-learning architecture to model relations between two simultaneous sequences of events, including temporal information about simultaneity and distances between events, without the need for time quantization or explicit musical macro-structural annotations. The model is designed to be trained on small, user-collected corpora of sound data and fine-tuned over repeated interactions for use in a live scenario. Our quantitative experiments demonstrate that the model achieves performance a canonical transformers while predicting descriptor values instead of discrete classes, and it can be reliably run in real time.

The main limitations of the system are related to corpus size and homogeneity. The model captures fundamental relations between instruments—for example, time-dependent relations between instrumental registers, tempo, and associations between types of sound materials—but it does not fully capture highly repetitive structures such as rhythms and chord progressions, probably because of the limited size of the context window. This could be solved using a multi-granular approach by encoding progressively longer time windows. The two musical case studies provided in this paper demonstrate ways in which a solo musician can collect interaction data. Further testing in practical application scenarios—ideally involving data collection from improvising duos—is necessary to fully evaluate the effectiveness of the model.

This model could be extended in several directions. Currently, it does not explicitly model meso-structure constraints typical of idiomatic music (e.g., rhythm and chords). These could be introduced by adding conditioning features such as beat position for rhythms, and chord-class labels for harmony, and/or quantizing tempo and pitches at inference. Other event-level features could be associated or substituted to impart performance-specific qualities that will condition the model's behaviors. This system can also be used to generate symbolic music by quantizing the output (e.g., to MIDI pitch classes) and by controlling the generation tempo in the live implementation. Finally, a natural next step is to extend the approach to model the relations among more than two simultaneous instrumental tracks, and to handle polyphony within a single track.

9 ETHICAL STANDARDS

This work was funded by the Department of Musicology at the University of Oslo as part of the first author's doctoral research fellowship. The research presented here relies on open-source software frameworks and publicly available data. We have released the code developed for this project as open-source software to ensure reproducibility. No human participants other than the authors were involved in the study. No Large Language Model was used while writing this paper. The authors report no conflicts of interest. The computations required for the machine-learning models used in this work are comparable to an individual's daily computer usage; therefore, the machine-learning component was not a major contributor to the project's environmental impact.

REFERENCES

Agarwal, S. and Sultanova, N. (2024). Music generation through transformers. *International Journal of Data Science and Advanced Analytics*, 6(6):302–306.

Allauzen, C., Crochemore, M., and Raffinot, M. (1999). Factor oracle: A new structure for pattern matching. In *International conference on current trends in theory and practice of computer science*, pages 295–310. Springer.

Assayag, G., Bonnasse-Gahot, L., and Borg, J. (2022). Cocreative interaction: Somax2 and the reach project. *Computer Music Journal*, 46(4):7–25.

Assayag, G. and Dubnov, S. (2004). Using factor oracles for machine improvisation. *Soft Computing*, 8(9):604–610.

Aucouturier, J.-J., Defreville, B., and Pachet, F. (2007). The bag-of-frames approach to audio pattern recognition: A sufficient model for urban soundscapes but not for polyphonic music. *The Journal of the Acoustical Society of America*, 122(2):881–891.

Ben-Tal, O. and Dolan, D. (2023). Musical and meta-musical conversations. In *AI Music Creativity (AIMC*. PubPub.

Benetatos, C., VanderStel, J., and Duan, Z. (2020). Bachduet: A deep learning system for human-machine coun-

terpoint improvisation. In *Proceedings of the International Conference on New Interfaces for Musical Expression*.

Bispo Junior, D. A., Leite, G. d. N. P., Droguett, E. L., de Souza, O. V. C., Lisboa, L. A., Cavalcanti, G. D. d. C., Ochoa, A. A. V., Costa, A. C. A. d., Vilela, O. d. C., Brennand, L. J. d. P., et al. (2025). Short-term wind power forecasting with transformer-based models enhanced by time2vec and efficient attention. *Energies*, 18(23):6162.

Blackwell, T., Bown, O., and Young, M. (2012). Live algorithms: Towards autonomous computer improvisers. In *Computers and creativity*, pages 147–174. Springer.

Bujard, B., Nika, J., Bevilacqua, F., and Obin, N. (2025). Learning relationships between separate audio tracks for creative applications. In *AI and Music Creativity (AIMC)*.

Carnovalini, F. and Rodà, A. (2020). Computational creativity and music generation systems: An introduction to the state of the art. *Frontiers in Artificial Intelligence*, 3:14.

Clarke, E. F. (1987). Levels of structure in the organization of musical time. *Contemporary music review*, 2(1):211–238.

Costanzo, R. (2025). Data knot - machine learning tools for low latency real-time performance. https://github.com/rconstanzo/data-knot. Library for Max/MSP; Accessed: 2026-02-13.

Dahlstedt, P. (2021). Musicking with algorithms: Thoughts on artificial intelligence, creativity, and agency. In *Handbook of artificial intelligence for music: Foundations, advanced approaches, and developments for creativity*, pages 873–914. Springer.

Déguernel, K., Vincent, E., and Assayag, G. (2018). Probabilistic factor oracles for multidimensional machine improvisation. *Computer Music Journal*, 42(02):52–66.

Diniz, P., Junior, D. A. D., Diniz, J. O., de Paiva, A. C., Silva, A. C. d., Gattass, M., Quevedo, R., Michelon, D., Siedschlag, C., and Ribeiro, R. (2022). Time2vec transformer: A time series approach for gas detection in seismic data. In *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*, pages 66–72.

Drummond, J. (2009). Understanding interactive systems. *Organised Sound*, 14(2):124–133.

Dubnov, S. (2021). Machine improvisation in music: Information-theoretical approach. In *Handbook of Artificial Intelligence for Music: Foundations, Advanced Approaches, and Developments for Creativity*, pages 377–408. Springer.

Dubnov, S., Assayag, G., Lartillot, O., and Bejerano, G. (2003). Using machine-learning methods for musical style modeling. *Computer*, 36(10):73–80.

Erdem, C., Wallace, B., and Jensenius, A. R. (2022). Cavi: A coadaptive audiovisual instrument–composition. In *International Conference on New Interfaces for Musical Expression*. PubPub.

Geng, D., Wang, B., and Gao, Q. (2023). A hybrid photovoltaic/wind power prediction model based on time2vec, wdcnn and bilstm. *Energy conversion and management*, 291:117342.

Ghisi, D. and Agostini, A. (2017). Extending bach: A family of libraries for real-time computer-assisted composition in max. *Journal of New Music Research*, 46(1):34–53.

Gifford, T., Knotts, S., McCormack, J., Kalonaris, S., Yee-King, M., and d'Inverno, M. (2018). Computational systems for music improvisation. *Digital Creativity*, 29(1):19–36.

Haki, B., Nieto, M., Pelinski, T., and Jordà Puig, S. (2022). Real-time drum accompaniment using transformer architecture.

Hakimi, S. H., Bhonker, N., and El-Yaniv, R. (2020). Bebopnet: Deep neural models for personalized jazz improvisations. In *21st International Society for Music Information Retrieval Conference (ISMIR)*, pages 828–836.

Jiang, N., Jin, S., Duan, Z., and Zhang, C. (2020). Rl-duet: Online music accompaniment generation using deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 710–718.

Jones, D. (2025). Signalflow: Explore sound synthesis and dsp with python. https://signalflow.dev/.

Karita, S., Chen, N., Hayashi, T., Hori, T., Inaguma, H., Jiang, Z., Someki, M., Soplin, N. E. Y., Yamamoto, R., Wang, X., et al. (2019). A comparative study on transformer vs rnn in speech applications. In *2019 IEEE automatic speech recognition and understanding workshop (ASRU)*, pages 449–456. IEEE.

Kazemi, S. M., Goel, R., Eghbali, S., Ramanan, J., Sahota, J., Thakur, S., Wu, S., Smyth, C., Poupart, P., and Brubaker, M. (2020). Time2vec: Learning a vector representation of time. *International Conference on Learning Representations*.

Madaghiele, V., Lisena, P., and Troncy, R. (2021). MINGUS: Melodic Improvisation Neural Generator Using Seq2Seq. In *22nd International Society for Music Information Retrieval Conference (ISMIR)*, Online.

Mateo, C. and Talavera, J. A. (2020). Bridging the gap between the short-time fourier transform (stft), wavelets, the constant-q transform and multi-resolution stft. *Signal, Image and Video Processing*, 14(8):1535–1543.

McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E., and Nieto, O. (2015). librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, volume 8, pages 18–25.

Minu, R., Nagarajan, G., Borah, S., and Mishra, D. (2022). Lstm-rnn-based automatic music generation algorithm. In *Intelligent and Cloud Computing: Proceedings of ICICC 2021*, pages 327–339. Springer.

Nichols, E. P., Kalonaris, S., Micchi, G., and Aljanaki, A. (2020). Modeling baroque two-part counterpoint with neural machine translation. *arXiv preprint arXiv:2006.14221*.

Nika, J., Déguernel, K., Chemla, A., Vincent, E., Assayag, G., et al. (2017). Dyci2 agents: merging the" free"," reactive", and" scenario-based" music generation paradigms. In *International Computer Music Conference (ICMC)*.

Nika, J., Schwarz, D., and Muller, A. (2025). Crafting musical agents: Interactive generation as a catalyst for artistic formalization. In *AI Music Creativity (AIMC)*.

Nuttall, T., Haki, B., and Jorda, S. (2021). Transformer neural networks for automated rhythm generation. In *NIME 2021*. PubPub.

Odegua, R. (2019). An empirical study of ensemble techniques (bagging, boosting and stacking). In *Proc. Conf.: Deep Learn. IndabaXAt*.

Panariello, C. and Déguernel, K. (2026). Otiac-co-improvising with a musical agent in a feedback-based guitar performance. In *New Interfaces for Musical Expression*.

Pasini, M., Nistal, J., Hayes, B., Bjare, M. R., Lattner, S., and Fazekas, G. (2026). Liveband: Live accompaniment generation in the audio domain. *arXiv preprint arXiv:2606.03803*.

Peñaloza, V. (2020). Time2vec embedding on a seq2seq bi-directional lstm network for pedestrian trajectory prediction. *Res. Comput. Sci.*, 149(11):249–260.

Pereira, I., Araújo, F., Korzeniowski, F., and Vogl, R. (2023). Moisesdb: A dataset for source separation beyond 4-stems.

Ragano, A., Benetos, E., and Hines, A. (2023). Learning music representations with wav2vec 2.0. In *2023 31st Irish Conference on Artificial Intelligence and Cognitive Science (AICS)*, pages 1–6. IEEE.

Rhyu, S., Choi, H., Kim, S., and Lee, K. (2022). Translating melody to chord: Structured and flexible harmonization of melody with transformer. *IEEE Access*, 10:28261–28273.

Roads, C. (2003). The perception of microsound and its musical implications. *Annals of the New York Academy of Sciences*, 999(1):272–281.

Roberts, A., Engel, J., Mann, Y., Gillick, J., Kayacik, C., Nørly, S., Dinculescu, M., Radebaugh, C., Hawthorne, C., and Eck, D. (2019). Magenta studio: Augmenting creativity with deep learning in ableton live. In *Proceedings of the international workshop on musical metacreation (mume)*, page 11.

Rowe, R. (1999). The aesthetics of interactive music systems. *Contemporary music review*, 18(3):83–87.

Scarlatos, A., Wu, Y., Simon, I., Roberts, A., Cooijmans, T., Jaques, N., Tarakajian, C., and Huang, A. (2025). Realjam: Real-time human-ai music jamming with reinforcement learning-tuned transformers. In *Proceedings of the Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, pages 1–9.

Selva, J., Johansen, A. S., Escalera, S., Nasrollahi, K., Moeslund, T. B., and Clapés, A. (2023). Video transformers: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(11):12922–12943.

Steinbeck, P. (2018). George lewis's voyager. In *The Routledge companion to Jazz studies*, pages 261–270. Routledge.

Su, J., Ahmed, M., Lu, Y., Pan, S., Bo, W., and Liu, Y. (2024). Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

Tatar, K. and Pasquier, P. (2017). Masom: A musical agent architecture based on self-organizing maps, affective computing, and variable markov models. In *Proceedings of the International Workshop on Musical Metacreation*, volume 10, page 5281.

Thelle, N. J. and Pasquier, P. (2021). Spire muse: A virtual musical partner for creative brainstorming. In *NIME 2021*. PubPub.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Veale, T. and Cardoso, F. A., editors (2019). *Computational Creativity - The Philosophy and Engineering of Autonomously Creative Systems*. Springer.

Vigliensoni, G., McCallum, L., Maestre, E., Fiebrink, R., et al. (2022). R-vae: Live latent space drum rhythm generation from minimal-size datasets. *Journal of Creative Music Systems*, 1(1).

Wang, C.-i. and Dubnov, S. (2014). Variable markov oracle: A novel sequential data points clustering algorithm with application to 3d gesture query-matching. In *2014 IEEE International Symposium on Multimedia*, pages 215–222. IEEE.

Wang, C.-I., Hsu, J., and Dubnov, S. (2016). Machine improvisation with variable markov oracle: Toward guided and structured improvisation. *Computers in Entertainment (CIE)*, 14(3):1–18.

Xenakis, I. (1971). *Formalized Music*. Indiana University Press, Bloomington.

Xiao, T., Xu, Z., He, W., Xiao, Z., Zhang, Y., Liu, Z., Chen, S., Thai, M. T., Bian, J., Rashidi, P., et al. (2025). Xtsformer: Cross-temporal-scale transformer for irregular-time event prediction in clinical applications. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 28502–28510.

Zhang, T., Ramakrishnan, R., and Livny, M. (1996). Birch: an efficient data clustering method for very large databases. *ACM sigmod record*, 25(2):103–114.