# A Loopy Framework and Tool for Real-time Human-AI Music Collaboration

**Sageev Oore**[1,4,*] **Finlay Miller**[1,4,*] **Chandramouli Sastry**[2†] **Sri Harsha Dumpala**[1,4]
**Marvin F. da Silva**[1,4] **Daniel Oore**[3] **Scott C. Lowe**[4]

[1]Dalhousie University [2]Amazon [3]Aix-Marseille University [4]Vector Institute [*]Joint contribution

osageev@gmail.com

https://osageev.github.io/projects/looper/

## Abstract

We propose that looping provides an especially effective framework for real-time human–AI musical collaboration. Within this setting, we introduce *SmartLooper*, a system designed to support improvisation through responsive and evolving loop generation. The musician first records a personal dataset of musical fragments. During performance, the musician initiates a starting point, and the system uses this to traverse the dataset via a stochastic process, selecting loop segments based on distances computed in an embedding space derived from a pretrained diffusion model. This enables smooth yet varied transitions, allowing the system to continually evolve while retaining the performer's stylistic identity. The musician can then layer new lines and textures over the evolving loop, creating a fluid and co-creative improvisation[1].

## 1 Introduction

**On finding the right fit.** A longstanding quote in the human-computer interaction community is that *every tool/design is best for something and worst for something else*, famously stated by Buxton [6, 7]. This also extends to interactive music generation; for example, for harmonic exploration, a piano is often a more natural fit than a drum kit. Conversely, if one wants to explore the relationship of breathing to phrasing, the piano is a much more challenging place to start than the human voice.

**Frameworks for human-AI music systems.** Multiple "musical frameworks" are possible for human-AI music interaction, each of which may be a more or less natural fit for the characteristics of the AI models and tasks involved. For example, one reason call-and-response [25, 5] frameworks work well for real-time human-AI interactions is because the overall flow is not adversely affected by the inherent unpredictability of the model. A good musician can "reply" quickly to a surprising musical statement, whereas trying to play along with it simultaneously would be much harder, and the flow would tend to fall apart. In a call-and-response format, even if the system was intended to respond to the user's musical phrase, when it does generate an entirely unexpected response (which these systems sometimes do), the format itself allows a good musician to immediately swap roles and respond to the AI's generated outputs, making it sound good through re-contextualization.

An example of a non–real-time musical framework is iterative construction or refinement. One system using this framework is Amuse [18], which was explicitly built to support an iterative co-composing

[†]Work done while at Dalhousie University and Vector.

[1]As with most music, the collaboration between a human musician and our SmartLooper system is best experienced directly rather than described. We therefore recommend first watching a short video here.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Artificial Intelligence for Music: Where Creativity Meets Computation.

process through rounds of multi-modal suggestions from the user and model-generated suggested chord progressions in response. Generally, interactive music-AI systems will be premised on tasks such as suggesting and controlling musical continuations [22, 16, 8], in-filling individual parts or harmonies [29, 9], harmonizing by voice leading [17], controlling by text (whether for generating musical material [8, 2] or for designing sounds [4]), and controlling by providing other modalities [11–13], to name a few. Pons et al. [24] provide a comprehensive survey of recent music created using AI and they detail the different ways that AI was used.

**Looping framework.** In this work, we present a simple framework—together with a method for working within it—that we have found to be a remarkably effective vehicle for *playing together in real time with a generative model*: looping.

Broadly, looping may be considered as a reflex between people and their environment: beginning with call-and-response and the use of echoic and reverberant space as an instrument—making sound that explores community, space, and one's own listening and sense of self. Over time, mechanical, then electrical, electronic, and digital means layered into these practices—pinned-cylinder automata (eg. music boxes), locked-groove discs, tape time-lag/loops, sampling, and today's multi-channel loopers—extending long-standing musical logics such as ostinati, phrase-length riffs, and cyclic rhythmic and harmonic forms. The loop is not static however; it evolves under muscular and respiratory gesture, acoustic response and decay, mechanism and medium drift, and the listener's shifting attention.

Some previous AI-based looping tools focused on providing offline editing tools to help produce precise, seamless loops [28, 3, 21]. While important non-AI systems like Live [1] are fundamentally designed to facilitate live performance with looping, the integration of impressive AI features within Live (e.g. Magenta [19]) has been supportive of elements other than the interactive looping process itself. By contrast, our system is intended explicitly for real-time interaction, and *an essential quality of our approach is that the system is not exactly repeating itself.* Instead, it evolves gradually, more like a human performer asked to maintain a groove while improvising: the loop deliberately develops and unfolds over time in a way that feels musical and engaging.

Among interactive AI-based looping systems, ReflexiveLooper [23, 20] assumes a harmonic grid (e.g. as in a jazz standard or pop song) and analyzes a guitarist's playing in real-time in order to synthesize audio of a guitar playing chords or basslines as played earlier by the musician, such that they will fit the current chord. In Living Looper [26, 27], the system attempts to fit a generative model to try to produce the sound of the recorded audio, and then uses that model to continue the sound; multiple such loopers are active and controllable simultaneously.

In contrast, SmartLooper is designed not for a fixed harmonic grid, but rather to support improvising over segments such as groove-length phrases: the groove may (and often does) change harmonically over time, but it doesn't change too fast. For example, the musician may start a groove in one or two chords, which the system picks up, and over time the system may add a new chord, or gradually add and remove notes that effectively alter a chord, but the important characteristic is that the pace allows a musician to improvise along with it, without a predetermined harmonic structure.

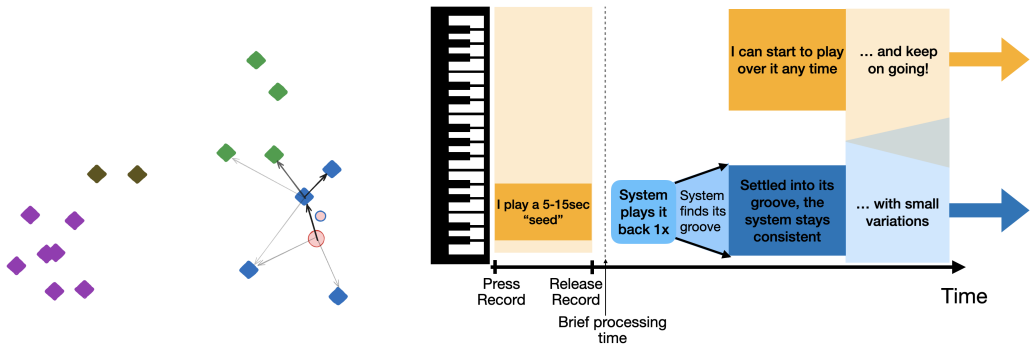
One reason looping is well-suited for a real-time, interactive AI system is that it is at its best with primarily *small variations* (feasible with AI techniques) between each loop, which lends itself well to being *predictable* for the human musician. Thus, the opportunity and challenge within the looping framework is to simultaneously allow the system to produce interesting variations, and also ensure those variations are semantically constrained enough that the system feels sufficiently predictable to play along with it in real-time. Notably, those same small variations might seem "boring" in a system not designed for looping, but in this context they are exactly what the task needs.

## 2 System Overview

Figure 1b shows a user's perspective of playing with SmartLooper.

### 2.1 Personal Dataset Collection (i.e. Creation)

The user first creates their own personal dataset of MIDI clips that they feel would be effective as looping material for this system. All clips are recorded with a metronome, to ensure they can be




(a) System traversing loop space

(b) User's view of the system

Figure 1: Schematic diagrams of two complementary perspectives on SmartLooper. (a) **The system's stochastic walk through loop segment space.** Each diamond is a recorded loopable segment in the dataset. The colours of the data segments represent the user's initial track-based annotation, e.g. all the blue segments were considered "similar" by virtue of having been played in a single looping track by the musician. The red circle represents the musician's seed recording. Distances are computed in the embedding space. Once the candidate neighbours are found, the next loop segment is identified (in this case the blue segment above). Darker arrows indicate more likely candidates. An interpolated segment is created during the warm-up (the smaller red circle with a blue outline). (b) **The user's view of the system.** Orange: **user** playing; blue: **AI system** playing. To start the process, the user records a 5–15 second seed, typically in the bass through middle pitches of the keyboard. The system plays back the seed, and then has a short warm-up phase during which it transitions to find its groove. Once it settles in, it stays robust and feels musically solid as it gradually shifts and evolves its groove. While we show the user and AI system mainly playing in upper and lower regions of the piano respectively (after the seed), there is no hard threshold between the two; the player is free to play in all areas of the piano, and the system may meander up or down the keyboard (though this is uncommon). The user can see on the computer screen the next segment queued from the looper before it is played, and plays over top of this continuous groove as they wish. All of these phases (including the user playing across the full keyboard) are shown in the demonstration video.

recombined and later easily time-stretched to fit together at different tempi. While not essential, the user can record tracks in which they loosely loop a groove, similarly to how they would like the system to play later. Doing so effectively provides an implicit annotation regarding the similarity of the segments in that track.

All clips were transposed by $[-18, -17, \cdots, +17, +18]$ semitones (up to 1.5 octaves), upsampling the dataset size by ×37. We denote the resulting augmented dataset $\mathcal{D}$. Importantly, note that *the artist who created the data retains ownership of it, and all the music generated by the system is derived directly from the data owned by the artist*.

### 2.2 Real-time System

Playing with SmartLooper consists of several key phases, described below.

**Phase 1: Launch.** The user launches the system with a single command and set of customizable options. These options include tempo (in beats per minute), loop match selection mode (see Appendix A), and other parameters relating to how the system should respond to what the musician is playing.

**Phase 2: Initial seed recording.** On startup, the system waits for the `record-start` signal (pressed with a foot pedal). This initiates an audible metronome, and the system will record (in MIDI) everything the musician plays until the `record-stop` signal is sent.This gives a seed recording, $\boldsymbol{s}$.

**Phase 3: Initializing loop.** The system uses seed $\boldsymbol{s}$, together with dataset $\mathcal{D}$, and distance function $\mathrm{d}(\cdot, \cdot)$ to find a small set of $N_{\text{init}}$ best initial matches $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N_{\text{init}}}\}$ which approximately minimize $\mathrm{d}(\boldsymbol{s}, \boldsymbol{x}_i)$ (Section A.1). It chooses from this set to determine the initial primary loop segment, $\boldsymbol{x}_{\text{init}}$ (different usage-modes/algorithms can lead to slightly different choices).

**Phase 4: Warm-up.** One detail we found to be helpful is adding a warm-up phase into the smart loop. Sometimes $\boldsymbol{x}_{\text{init}}$ seems perceptually a bit "far" from $\boldsymbol{s}$. To adjust for this, instead of going straight from $\boldsymbol{s}$ to $\boldsymbol{x}_{\text{init}}$, the system begins by repeating an excerpt $\boldsymbol{s}'$ of $\boldsymbol{s}$, which is close to $\boldsymbol{x}_{\text{init}}$ and which it finds to be well-suited for looping. It then follows $\boldsymbol{s}'$ with a few simple interpolations to reach $\boldsymbol{x}_{\text{init}}$ (i.e. adding or removing a few notes), computed to roughly minimize distances $\mathrm{d}(\boldsymbol{s}, \boldsymbol{s}'), \mathrm{d}(\boldsymbol{s}', \boldsymbol{s}''), \mathrm{d}(\boldsymbol{s}'', \boldsymbol{x}_{\text{init}})$, where each segment in the sequence $\{\boldsymbol{s}, \boldsymbol{s}', \boldsymbol{s}'', \ldots\}$ is obtained by modifying the previous segment. Since the warm-up is short and the interpolations are not refined, this can lead to brief audible glitches; in future version we will address this.

**Phase 5: Markov process.** After the warm-up, the system has played sample $\boldsymbol{x}_{\text{init}}$ and entered the graph of nodes within $\mathcal{D}$. From now on, we sample the next segment to play from $\mathcal{D}$ using a Markov process, described below. Figure 1a illustrates the general approach, and more details are provided in Appendix A.

**Additional Control Interface.** While the above process runs, in addition to an indirect control afforded by a player tracking functionality (described below), the system also supports the use of a second external MIDI controller. This allows the user straightforward, direct control over several parameters of the playback including volume, player tracking sensitivity, and transposition.

## 3 Results

We recommend watching a short demonstration video to see the system in action. While objective evaluation is challenging, we note that this video was not cherry-picked; we selected this sample from a small pool of recordings because it was highly representative of the system. It illustrates, within a 2-minute period, many typical characteristics: the close relationship between the original seed and the resulting groove, the glitches during the 10-second warm-up period, the robustness and rhythmic and harmonic stability of the groove after that, the musician occasionally playing in the bass, the gradual but clear evolution of the loop over time, and the genuine playfulness that it allows (the musician truly had a fun time during the entire session!). In addition to the accompanying demonstration video, other grooves during that same session ranged from slow and sparse ballad-like loops to Cuban montunos. The system is robust enough that a musician has performed with the system successfully in front of live audiences.

## 4 Discussion

In this work, we have described the advantages of a looping framework for exploring human-AI music collaboration. We presented a working, effective, and personable interactive system which creates dynamically changing musical loops. The changes to the loop generated by our system are sufficiently consistent that it does not make semantically large jumps in its content and thus can be accompanied by a human player in real-time.

## Acknowledgements

We thank the anonymous reviewers for their feedback.

Resources used in preparing this research were provided, in part, by the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute.

## References

[1] Ableton. Live 12. Commercial Software, 2025. URL https://www.ableton.com/.

[2] Andrea Agostinelli, Timo I. Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Caillon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, Matt Sharifi, Neil Zeghidour, and Christian Frank. Musiclm: Generating music from text, 2023. URL https://arxiv.org/abs/2301.11325.

[3] Audjust. AI analysis features for looping. Commercial Software, 2025. URL https://www.audjust.com/.

[4] Stephen Brade, Bryan Wang, Mauricio Sousa, Gregory Lee Newsome, Sageev Oore, and Tovi Grossman. SynthScribe: Deep multimodal tools for synthesizer sound retrieval and exploration. In *Proceedings of the 29th International Conference on Intelligent User Interfaces*, IUI '24, page 51–65, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400705083. doi:10.1145/3640543.3645158.

[5] Mason Bretan, Sageev Oore, Jesse Engel, Douglas Eck, and Larry Heck. Deep music: Towards musical dialogue. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1), Feb 2017. doi:10.1609/aaai.v31i1.10544.

[6] Bill Buxton. Multi-touch systems that I have known and loved. online, 2007. URL https://www.billbuxton.com/multitouchOverview.html.

[7] Bill Buxton. *Sketching User Experiences: Getting the Design Right and the Right Design (Interactive Technologies)*. Morgan Kaufmann, 2007.

[8] Antoine Caillon, Brian McWilliams, Cassie Tarakajian, Ian Simon, Ilaria Manco, Jesse Engel, Noah Constant, Pen Li, Timo I. Denk, Alberto Lalama, Andrea Agostinelli, Anna Huang, Ethan Manilow, George Brower, Hakan Erdogan, Heidi Lei, Itai Rolnick, Ivan Grishchenko, Manu Orsini, Matej Kastelic, Mauricio Zuluaga, Mauro Verzetti, Michael Dooley, Ondrej Skopek, Rafael Ferrer, Zalán Borsos, Äaron van den Oord, Douglas Eck, Eli Collins, Jason Baldridge, Tom Hume, Chris Donahue, Kehang Han, and Adam Roberts. Live music models. *arXiv:2508.04651*, 2025. doi:10.48550/arXiv.2508.04651.

[9] Chris Donahue, Shih-Lun Wu, Yewon Kim, Dave Carlton, Ryan Miyakawa, and John Thickstun. Hookpad Aria: A copilot for songwriters. In *Extended Abstracts for the Late-Breaking Demo Session of the 25th International Society for Music Information Retrieval Conference*, 2024.

[10] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The Faiss library. 2024. doi:10.48550/arXiv.2401.08281. URL https://arxiv.org/abs/2401.08281.

[11] Jason d'Eon, Sri Harsha Dumpla, Chandramouli Shama Sastry, Daniel Oore, and Sageev Oore. Musical speech: a transformer-based composition tool. In *NeurIPS 2020 Competition and Demonstration Track*, pages 253–274. PMLR. URL https://proceedings.mlr.press/v133/d-eon21a.

[12] Rebecca Fiebrink and Perry R Cook. The wekinator: a system for real-time, interactive machine learning in music. In *Proceedings of The Eleventh International Society for Music Information Retrieval Conference (ISMIR 2010)(Utrecht)*, volume 3, pages 2–1, 2010.

[13] Hugo Flores García, Oriol Nieto, Justin Salamon, Bryan Pardo, and Prem Seetharaman. Sketch2sound: Controllable audio generation via time-varying signals and sonic imitations, 2025. URL https://arxiv.org/abs/2412.08550.

[14] Josh Gardner, Ian Simon, Ethan Manilow, Curtis Hawthorne, and Jesse Engel. MT3: Multi-task multitrack music transcription, 2022. URL https://arxiv.org/abs/2111.03017.

[15] Curtis Hawthorne, Ian Simon, Adam Roberts, Neil Zeghidour, Josh Gardner, Ethan Manilow, and Jesse Engel. Multi-instrument music synthesis with spectrogram diffusion, 2022. URL https://arxiv.org/abs/2206.05408.

[16] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Curtis Hawthorne, Andrew M Dai, Matthew D Hoffman, and Douglas Eck. Music transformer: Generating music with long-term structure. *arXiv preprint arXiv:1809.04281*, 2018. doi:10.48550/arXiv.1809.04281.

[17] Cheng-Zhi Anna Huang, Tim Cooijmans, Adam Roberts, Aaron Courville, and Douglas Eck. Counterpoint by convolution, 2019. URL https://arxiv.org/abs/1903.07227.

[18] Yewon Kim, Sung-Ju Lee, and Chris Donahue. Amuse: Human-AI collaborative songwriting with multimodal inspirations, 2025. URL https://arxiv.org/abs/2412.18940.

[19] Google Magenta. Studio. Ableton Live Plugin, 2025. URL https://magenta.withgoogle.com/studio.

[20] Marco Marchini, François Pachet, and Benoît Carré. Rethinking Reflexive Looper for structured pop music. *NIME*, 2017.

[21] Davide Marincione, Giorgio Strano, Donato Crisostomi, Roberto Ribuoli, and Emanuele Rodolà. LoopGen: Training-free loopable music generation, 2025. URL https://arxiv.org/abs/2504.04466.

[22] Nicholas Meade, Nicholas Barreyre, Scott C. Lowe, and Sageev Oore. Exploring conditioning for generative music systems with human-interpretable controls. In *International Conference on Computational Creativity (ICCC)*, 2019. doi:10.48550/arXiv.1907.04352. URL https://computationalcreativity.net/iccc2019/papers/iccc19-paper-57.pdf.

[23] François Pachet, Pierre Roy, Julian Moreira, and Mark d'Inverno. Reflexive loopers for solo musical improvisation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2205–2208, Paris France, April 2013. ACM. ISBN 978-1-4503-1899-0. doi:10.1145/2470654.2481303.

[24] Jordi Pons, Zack Zukowsi, Julian D. Parker, CJ Carr, Josiah Taylor, and Zach Evans. Music and artificial intelligence: Artistic trends, 2025. URL https://arxiv.org/abs/2508.11694v1.

[25] Adam Roberts, Jesse Engel, Curtis Hawthorne, Ian Simon, Elliot Waite, Sageev Oore, Natasha Jaques, Cinjon Resnick, and Douglas Eck. Interactive musical improvisation with Magenta. In *Advances in Neural Information Processing Systems*, 2016.

[26] Victor Shepardson and Thor Magnusson. The living looper: Rethinking the musical loop as a machine action-perception loop. In *Proceedings of the International Conference on New Interfaces for Musical Expression*. URL http://nime.org/proceedings/2023/nime2023_32.pdf.

[27] Victor Shepardson, Halla Steinunn Stefánsdóttir, and Thor Magnusson. Evolving the Living Looper: Artistic Research, Online Learning and Tentacle Pendula. *NIME*, 2025.

[28] Soundverse. Auto-loop feature. Commercial Software, 2025. URL https://www.soundverse.ai/music-looper-ai.

[29] John Thickstun, David Hall, Chris Donahue, and Percy Liang. Anticipatory music transformer, 2024. URL https://arxiv.org/abs/2306.08620.

# A Markov Process over Loop Segments

## A.1 Computing Distances Between Loop Segments

Once a dataset has been collected and a seed segment recorded, the key problem is selecting the next loop to play. Formally: given a set of loopable segments, how do we find the ones that are musically closest to the current segment? This can be tricky; for example, in some contexts the distance from E to E-flat (one semitone) may feel larger than the distance from E to B (a perfect fifth), while in others the reverse holds. After identifying a good set of neighbours, the system can choose one to play next. We initially tested hand-crafted features (e.g., pitch histograms, piano-roll descriptors), but each of these produced occasional abrupt transitions that were disorienting for both musician and audiences.

Our approach here instead computes distances in the embedding space of a pretrained MIDI-to-audio diffusion model. Specifically, for each MIDI segment $\boldsymbol{x}_i$ we obtain a 768-dimensional embedding $\boldsymbol{e}_i$ using the MT3-based [14] note encoder from Hawthorne et al. [15]. We then compute distances for segments $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ as their cosine distance in embedding space:

$$\mathrm{d}(i,j) = 1 - \frac{\boldsymbol{e}_i \cdot \boldsymbol{e}_j}{\|\boldsymbol{e}_i\| \, \|\boldsymbol{e}_j\|}.$$

This captures salient musical properties more effectively, enabling smoother loop transitions. Having defined this, we support several different methods for choosing which loop to play next.

## A.2 Deterministic Walks

In the simplest method, the system selects the nearest unplayed neighbour, $\boldsymbol{x}'_i$, to the segment it is currently playing, $\boldsymbol{x}_i$ based on cosine distance. This yields smooth transitions, though it can sometimes stay locked in one groove for too long.

Because the method is deterministic, it can be viewed as a degenerate Markov chain with one-hot transition probabilities: transitions follow the heuristic $f(i) = \arg\min_{j \notin R} \mathrm{d}(i,j)$, where $R$ is the set of most recently played clips. This path can be precomputed after the first step $\boldsymbol{s} \to \boldsymbol{x}_{\text{init}}$. To run in real time, we use FAISS [10] for $k$-nearest neighbor search over the loop segment embeddings, applying a filter to avoid recently played files.

## A.3 Player Tracking

A player tracking mode can respond to the musician's live input by calculating embeddings of what the musician just played, and using those to influence the similarity search being run by the system during performance. The interface easily allows the musician switch between a few different modes of doing this, as well as a continuous weighting parameter to control the strength of the tracking, but we found that, overall, this made the system harder and less satisfying to use by injecting too much unpredictability.

## A.4 Stochastic Transitions

To encourage more dynamic behaviour without losing local smoothness, we introduce probabilistic transitions. Several variants are possible: for example, the system can periodically force a jump to a different loop, assign probabilities that decay with embedding-space distance, or combine both approaches. In all cases, the resulting quality critically depends on the embedding distance: without a musically meaningful similarity measure, larger leaps quickly become incoherent.

We define a discrete distribution $P$ over a set of transition options $\Omega$, grouped into three categories: (1) top-$k$ similar segments (as in Section A.2); (2) top-$k$ similar segments among those annotated by the user as being similar to the current segment; (3) top-$k$ similar segments among those not annotated by the user as being similar. The probabilities assigned to these categories are tunable by the musician, either before or during performance, allowing them to control the dynamics of the system's trajectory.