California Institute of the Arts

Loops That Listen: A Voice-Controlled Dynamic Drum Looper with AI Variation

A thesis submitted in partial satisfaction

of the requirements for the degree Master of Arts

in Aesthetics and Politics

by

Paolo Sandejas

2026

Table of Contents

List of Table and Figures

Abstract

Live looping constrains musical dynamics. The same pattern cycles indefinitely, while a performance demands response and variation. CHULOOPA is a real-time drum looper that resolves this tension directly. The performer beatboxes patterns, and the system varies them in response to the live musical energy. A dBFS-mapped RMS measure, “spice”, drives selection from a pre-generated bank of five variations. These variants are generated at different temperatures in parallel by a grid-based decoder-only transformer and then sorted by deviation score to maintain a predictable creativity spectrum for the performer. At each loop boundary, the variant matching the current spice level is selected without interrupting playback. CHULOOPA demonstrates that the loop’s inherent repetition need not be a creative constraint. With audio-responsive AI variation, a loop becomes dynamically expressive without displacing the performer’s musical intent.

Introduction

The strength of a loop is also its main constraint: once recorded, it never changes. For drum loops in live performance, this constraint is especially pronounced. Where a live drummer would subtly add fills, vary dynamics, or shift the groove to match the moment, a drum loop plays back identically every cycle. While not always undesired, with hardware loop pedals like the BOSS RC-1 and software tools such as Ableton Live's Session View excelling at stacking static loops to provide rhythmic stability to a performance, an unchanging loop lacks the organic variation that makes live drumming compelling. With that limitation in mind, how can dynamic musicality be brought into drum loops without sacrificing their stability?

CHULOOPA (ChucK-based (Wang et al., 2015) Loop Operator for Performance Audio) is a voice-controlled dynamic drum looper that addresses this. Unlike prior intelligent loopers (Marchini et al., 2017; Burloiu, 2020) that work from guitar audio or pre-loaded MIDI drum scores, CHULOOPA uses voice input as its sole creation interface: a performer beatboxes a drum pattern, the system transcribes it in real-time via a user-trained KNN classifier, loops it, and immediately generates a bank of 5 AI variations at different creative levels. Simultaneously, the variant controller reads the live audio energy, spice, and at each loop boundary performs a weighted probabilistic selection from the variation bank to match the current musical moment, without manual intervention.

Chapter 1: Related Work

1.1 Intelligent Loopers

From Pachet's Continuator (Pachet, 2003) and OMax (Assayag et al., 2006), which pioneered real-time style learning and pattern recombination from live input, to the Reflexive Looper (Marchini et al., 2017), which established looping agents that adapt to a musician using constrained optimization to schedule material across upcoming beats, intelligent loopers have pursued the same goal: making loops responsive to a live performer. Rolypoly~ (Burloiu, 2020) takes a complementary approach, using RNNs pretrained on drum groove data to adapt the microtiming and velocity of a loaded drum score to a musician's timing, modulating how a pattern is played. The Living Looper (Shepardson & Magnusson, 2023; Shepardson et al., 2025) extends this lineage: rather than replaying recorded audio, each loop channel uses a pre-trained RAVE autoencoder to project the recording into a latent space, where a lightweight linear model is fit per-recording, allowing the loop to drift and transform over time. Madaghiele et al. (2025) push this toward full autonomy: their Multi-Agent Autonomous Looper (MAAL) assigns a rule-based agent to each loop track; agents continuously evaluate the rhythmic and harmonic compatibility of incoming live segments and autonomously decide what to loop, removing manual record triggers entirely. Recursive Speculation (Madaghiele & Cotton, 2025) demonstrates this with vocal input, where MAAL autonomously samples and layers a performer's improvised extended techniques into evolving looped structures.

1.2 AI Drum Variation

Nuttall et al. (2021) first trained a Transformer-XL on the Groove MIDI Dataset for drum sequence generation; Haki et al. (2022) subsequently demonstrated real-time Transformer-based drum accompaniment in performance contexts. Evans et al. (2024) extended this line of work into the GrooveTransformer, a VAE-based Eurorack module for real-time generative drum sequencing. GrooveTransformer steers variation through two pre-selected latent anchors and live MIDI input; it does not derive selection from raw audio energy, and generation is bounded by those reference points rather than a single live-recorded loop. Bruford et al. (2020) present jaki, which generates 1-bar LSTM continuations of a seed drum pattern and tunes them to user-defined musical features (syncopation, density, repetition) via deep reinforcement learning. Alain et al. (2020) use active learning to converge on user drum-loop preferences from like/dislike ratings. Brosnan (2026) presents DARC, a drum accompaniment model that conditions on musical context and a rhythm prompt (e.g. beatboxing), targeting music prototyping rather than live performance.

1.3 Beatbox Transcription

Mapping vocal percussion to drum classes is a cross-modal transcription task: Santos et al. (2025) formalize it as mapping arbitrary percussive inputs to kick/snare/hat labels. Delgado et al. (2019) showed that vocal percussion styles vary significantly across amateur performers, and Ramires et al. (2018) demonstrated that personalized, user-specific classifiers substantially outperform generic models because different users

produce acoustically distinct vocal percussion for the same drum class. Martanto & Kartowisastro (2025) confirm that KNN with MFCC features achieves strong accuracy for voice-derived percussion. Under live performance latency constraints, Stowell & Plumbley (2010) demonstrated real-time beatbox classification using delayed decision-making, and Weber et al. (2024) showed that high-quality drum transcription is achievable with minimal training data. Together, this body of work prioritizes personalization and real-time latency over generalization.

Chapter 2: System Design

CHULOOPA runs as three concurrent processes, illustrated in Figure 1: the main looper handles beatbox recording, drum classification, loop playback, and variation selection; the variant generator generates a bank of five AI variations in the background the moment a new pattern is recorded; and the variant controller continuously monitors the live audio environment, streaming a single spice value every 500 ms. Separating spice detection into its own process was a deliberate design choice. Both the main looper and variant controller process live audio, but for different purposes: drum onset classification in the former and dBFS energy measurement in the latter.

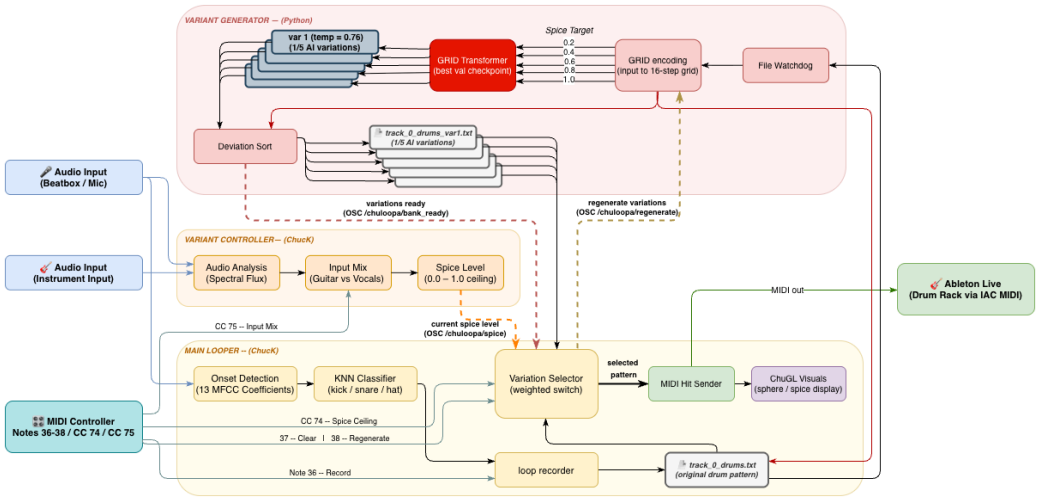

Figure 1. CHULOOPA system architecture. Three concurrent processes communicate via OSC: main looper, variant generator, and variant controller.

2.1 Beatboxing Classifier

To initialize CHULOOPA, the performer first teaches the system their own beatboxing vocabulary. Rather than relying on a generic drum-sound dataset, CHULOOPA asks the performer to record 10 examples each of kick, snare, and hi-hat, creating a small personalized classifier that adapts to the performer's vocal timbre and articulation in under two minutes. The system extracts 13 MFCC coefficients per onset, a standard count for short audio classification tasks (Martanto & Kartowisastro, 2025). A KNN classifier (Cover & Hart, 1967) ($k = 3$), implemented via ChAI's KNN2 object (Li & Wang, 2024), trains automatically at startup in under one second. KNN was chosen deliberately: it performs well with as few as 30 samples, trains instantly, and requires no hyperparameter tuning. User-specific training constrains the problem space enough that a simple classifier suffices.

2.2 Real-Time Transcription

Immediate audio feedback during recording is central to CHULOOPA's performance model: the performer hears classified drum samples in real-time while beatboxing, allowing technique correction before committing a pattern to loop. This requires reliable onset detection and sub-30 ms classification latency. Onset detection uses spectral flux with adaptive thresholding (1.5× running mean) in 512-sample frames, 128-sample hop (75% overlap). The 512-sample frame provides 86 Hz frequency resolution at 44.1 kHz, necessary to discriminate kick fundamentals (~60–100 Hz) from mid-frequency snare energy. Each onset is classified (kick/snare/hat) via MFCC-13 KNN

and the corresponding drum sample plays immediately. Simultaneously, a MIDI note-on is sent to Ableton Live via the macOS Inter-Application Communication (IAC) driver, routing hits to a Drum Rack and giving the performer full control over drum sound design independently of the classification engine. Total input-to-playback latency is ~25 ms. Each hit is stored with delta-time encoding—the interval to the next hit or to loop end for the final hit—enabling drift-free loop playback. For AI variation, this timestamped sequence is quantized to a 16-step grid and written back as the active playback loop; the quantized version is then passed to the GRID model, ensuring the original and all variations share the same temporal reference.

2.3 Spice Detection

To autonomously select from the variation bank, spice is defined as a normalized measure of live audio energy that controls the selection of generated variations. The variant controller continuously measures the energy of the live performer and maps it to a spice value from 0.0 to 1.0, allowing CHULOOPA to autonomously respond to changes in musical intensity. RMS energy was chosen as the primary signal because it captures the cumulative acoustic energy of the room, with no additional sensing required. Quieter passages favor conservative patterns; building energy pulls the system toward more adventurous ones. The detector averages the RMS energy over a 500 ms window and converts to a dBFS scale: spice = clamp((dBFS − F) / (P − F), 0.0, 1.0), where F is the noise-floor threshold (−75 dBFS) and P is the expected peak performance level (−50 dBFS). The dBFS mapping was chosen deliberately: a linear RMS scale would compress

most of the musically-useful dynamic range near zero, whereas loudness perception is logarithmic. In performance mode, with a two-channel audio interface, the input mix knob blends the instrument and vocal inputs before analysis, so spice reflects the collective energy of the full performance rather than the beatboxer's voice alone. The 500 ms window smooths over transient spikes while remaining responsive to sustained shifts in musical energy. Spice is streamed to the main looper process via OSC every 500 ms.

2.4 AI Variation Bank

CHULOOPA's generation engine is a grid-based decoder-only transformer (GRID), a 6-layer model with 256-dim embeddings, 8 attention heads, feed-forward dimension 1024, 42-token vocabulary, and 4.8M parameters, trained on the Expanded Groove MIDI Dataset (Callender et al., 2020). Bar pairs are constructed from performances containing at least two bars; pairs with empty bars or adjacent duplicates are discarded, yielding 12,085 training pairs. The model is conditioned on bar pairs: given a context bar, it generates the immediately following bar. A masked loss is applied only to the continuation bar, training the model to predict the next bar rather than reconstruct the context. Patterns are quantized to a 16-step grid and encoded as alternating P{step}/N{pitch} tokens; generation terminates at a learned end-of-sequence marker. When a loop is recorded, the variant generator immediately launches five parallel generation threads, one per spice target (0.2, 0.4, 0.6, 0.8, 1.0), chosen to maximize coverage of the spice spectrum while keeping generation overhead tractable; finer granularity would produce negligible perceptual difference between adjacent variants.

Each thread feeds the recorded pattern as a GRID token sequence to the model, generating a new bar conditioned on the input. Each thread is assigned a sampling temperature that increases with its spice target ($T = 0.6 + 0.8 \times \text{spice}$, giving $T = 0.76$ for the lowest-target thread and $T = 1.40$ for the highest), biasing each toward outputs of different adventurousness without guaranteeing it. A post-generation deviation sorting step enforces the final conservative-to-adventurous ordering. Because temperature steers tendency, not outcome, a high-temperature run may occasionally produce a more conservative variation. To ensure the spice axis is musically reliable, the five generated variants are sorted ascending by the tuple (Δhits, Vnew), where Δhits is the signed hit count difference from the original and Vnew is the count of drum voice classes absent from the original (Gillick et al., 2019). Density increase is the primary criterion; new voice classes break ties within the same density bucket. Sparser variations are not penalised. After sorting, slot 1 always holds the most conservative result and slot 5 the most adventurous, independent of which thread generated it. Total generation time: 3–5 seconds on consumer CPU hardware, entirely offline.

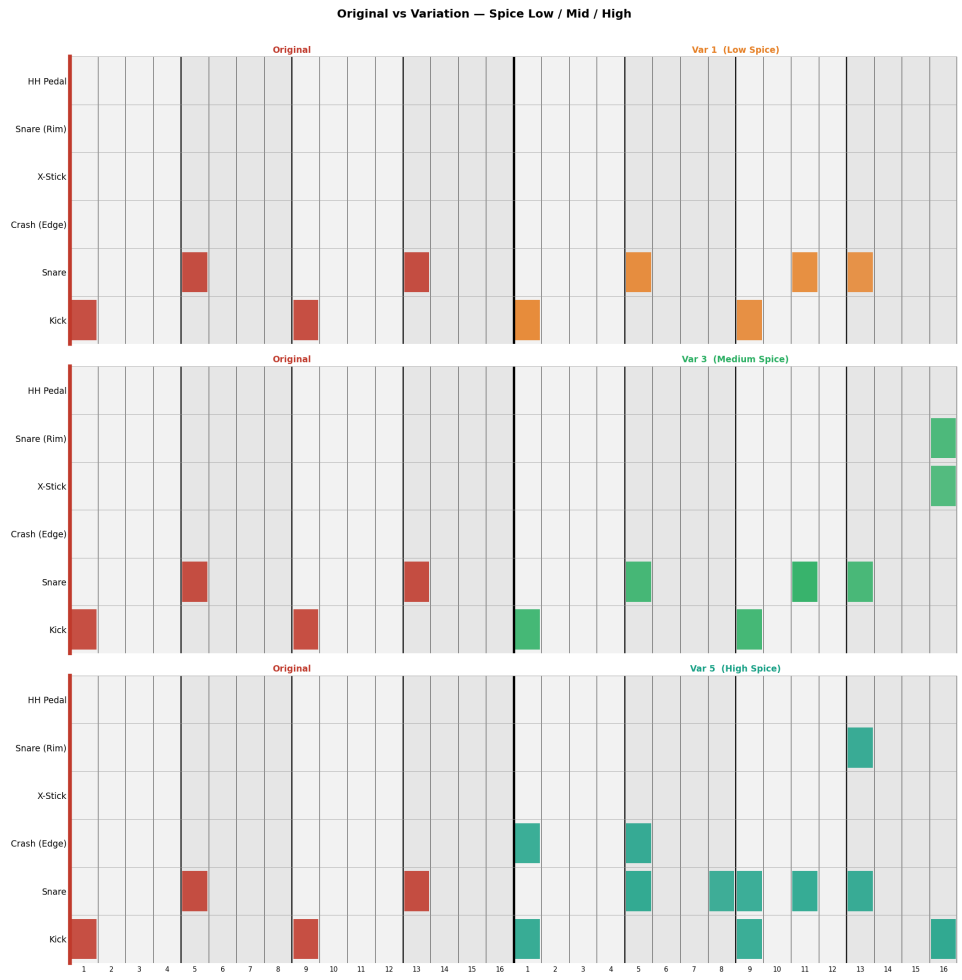

Figure 2. Drum grid comparison showing the original recorded pattern alongside AI-generated variations at low (Var 1), medium (Var 3), and high (Var 5) spice levels. Higher spice produces denser fills and introduces new drum voices.

2.5 Weighted Variation Selection

At each loop boundary, the main looper draws from a weighted distribution that shifts toward higher-numbered variants as spice rises (Table 1). The distribution includes the original pattern as a valid option: at low spice the system may simply continue playing the original, treating restraint as a musical choice. Selection uses a rolling 4-bar spice average, with no variant playing more than twice consecutively. The live performer

retains creative control through a configurable spice ceiling, mapped to CC 74. Rather than selecting variations manually, the performer decides how much latitude to give the system to respond to the music accordingly.

Table 1. Selection weights per spice tier.

Spice	Orig.	V1	V2	V3	V4	V5
0.00	0.50	0.40	0.10	0.00	0.00	0.00
0.25	0.20	0.40	0.30	0.10	0.00	0.00
0.50	0.05	0.20	0.40	0.30	0.05	0.00
0.75	0.00	0.10	0.30	0.40	0.20	0.00
1.00	0.00	0.00	0.20	0.40	0.30	0.10

2.6 Live Performance Controls

Inspired by hardware loopers, variation loading queues during the current cycle and executes at the next loop boundary, preventing mid-pattern interruptions. Each playback session holds a unique ID; scheduled drum hits validate their session before firing, eliminating overlap during transitions. MIDI controls: Note 36 (hold) to record loop; Note 37 to clear; Note 38 to regenerate full variation bank; CC 74 to set spice ceiling (caps audio-driven selection); CC 75 to adjust instrument/vocal input mix. ChuGL (Aday & Wang, 2024) visual feedback (Figure 3) closes the loop for the performer: the central shape shifts color with the active variation slot (blue = Var 1 through red = Var 5) and pulses on each hit, while a spice slider reflects the instantaneous spice level. The color is discrete, updating at each loop boundary; shape geometry tracks live spice,

shifting from a cube at low energy to an icosahedron at peak. A pulse fires at the moment each beatbox onset is classified, letting the performer confirm each hit in real time.

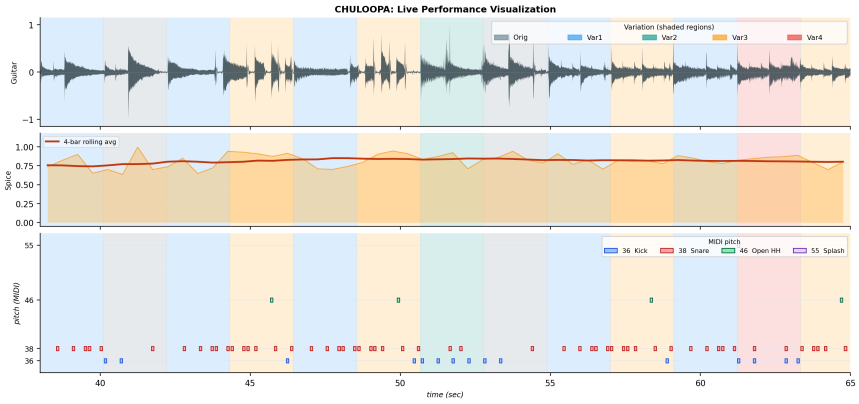

Figure 3. CHULOOPA performance UI. Shape colour tracks spice level (blue = low, yellow = high); spice slider shows the rolling average currently driving variation selection.

Chapter 3: Evaluation

The central evaluation question is whether CHULOOPA's AI-generated variations remain musically coherent with the human performance while offering a reliable, controllable range of complexity.

3.1 Variation Coherence

Variation coherence is evaluated by running 10 independent generation banks on a two-kick, two-snare pattern spanning 2.6 seconds. Each bank contains five variants at spice levels 0.2–1.0, deviation-sorted so slot 1 is always the most conservative result and slot 5 the most adventurous. Hit-probability heatmaps (Figure 4) aggregate binary grids across all 10 runs; darker cells indicate more consistent activation.

At low spice, Var 1 produces an average of 5.4 hits/bar, with 82.5% of input cells activating consistently, confirming strong input preservation. Additional voices appear at low probability (HH Pedal, Crash 2 Edge, Ride Edge), representing occasional idiomatic ornamentation rather than structural additions. At high spice, Var 5 reaches 12.7 hits/bar: the snare row spreads broadly, the kick maintains its step-1 anchor, and hi-hat and tom voices emerge. Crucially, additions concentrate on a small fraction of non-input positions: averaged across all empty grid cells, activation probability remains below 4% even at high spice (0.7% at Var 1, 3.1% at Var 5), confirming the model targets musically plausible positions rather than filling the grid at random. The density gradient (5.4→ 12.7) confirms the spice axis produces a reliable complexity spectrum: the system generates sparser outputs at low spice and more elaborated ones at high spice, with input positions serving as structural anchors throughout.

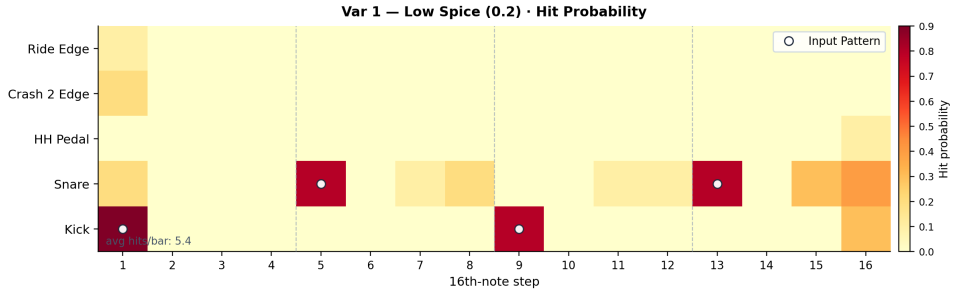

Figure 4 (top). Var 1 (Low Spice, 0.2) — sparse additions, input positions dominant. Aggregated over N=10 independent banks; white circles mark input pattern positions.

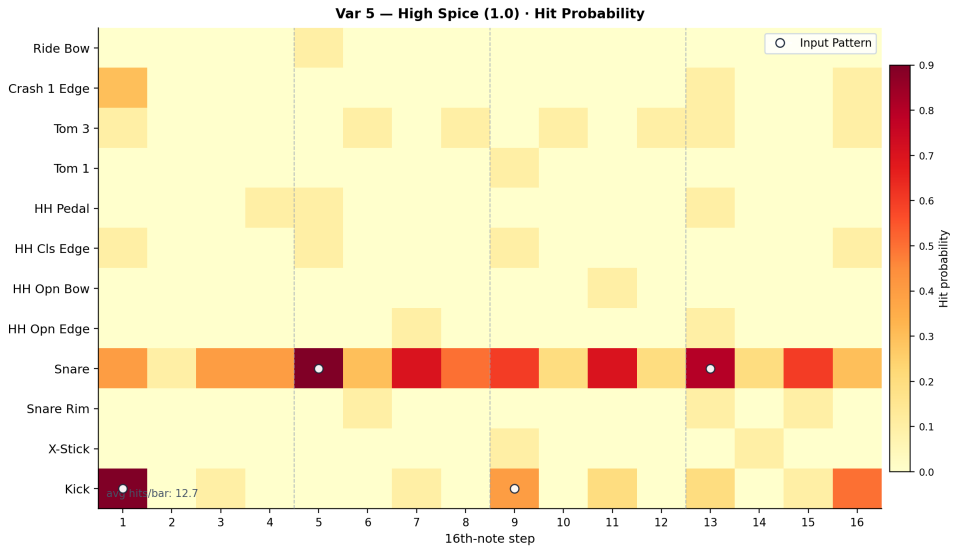

Figure 4 (bottom). Var 5 (High Spice, 1.0) — elaborated groove with hi-hats, toms, and snare fills; input anchors remain visible at steps 1 and 9.

3.2 Aesthetic Observations

The first author performed with CHULOOPA across multiple solo sessions, each lasting 10–20 minutes. All generated variations were assessed as coherent musical ideas; the only failure cases could be traced to input-side quantization artifacts rather than output-side incoherence. The strict 16-step grid eliminates beat-drift entirely. Mid-range spice values (approximately 0.4–0.6) proved most musically satisfying, neither too sparse nor too elaborate. Hearing the system return to the original pattern at low spice served as a rhythmic anchor, reinforcing awareness of the source groove. Unlike static loops, which can grow stale when brought to the forefront of the performance, CHULOOPA invites attention when given the spotlight. When holding a chord or leaving space in the arrangement, the variations shine as a musical statement in their own right, with the performer remaining compositionally in control.

Figure 5. CHULOOPA performance screenshot with UI samples from variation playback (left) and echo playback (right) - full video: https://sites.google.com/view/chuloopa/home

Conclusion

CHULOOPA demonstrates that AI-generated variation can bring dynamic musicality to the drum loop without sacrificing its stability. At low spice, 82.5% of original hit positions activate consistently across N = 10 independent banks, confirming the model treats the performer's pattern as a structural anchor rather than a seed to discard. Deviation-sorted parallel generation produces a reliable density gradient, 5.4 to 12.7 hits/bar from slot 1 to slot 5, confirming the spice axis as a controllable complexity spectrum. Spice drives autonomous selection at each loop boundary, matching the

variation to the current musical energy without demanding performer intervention. An AI looper can elaborate a groove without displacing it.

Current constraints are deliberate tradeoffs. The 16-step quantization grid is the source of CHULOOPA's temporal reliability, but it excludes swing feel, non-4/4 meters, and 32nd-note micro-timing. Variations are conditioned to continue from the input rather than replicate it, always coherent with the original but free to reinterpret it. Spice tracks amplitude alone, a reliable proxy for musical energy but blind to harmonic or rhythmic context. The evaluation is self-reported by the author; a structured listener study would strengthen the coherence claims.

Natural extensions include multi-track looping with coherent cross-track variation, swing and compound-meter support, and melodic generation. An open question is whether spice can be enriched by incorporating spectral complexity and rhythmic density alongside amplitude, and how drum patterns are expected to adapt to those changes, giving the selection engine a fuller picture of when variation is musically called for. The variation bank and autonomous selection together show that the loop's repetition can be its starting point, not its ceiling.

Works Cited

Aday, A. Z., & Wang, G. (2024). “ChuGL: Unified audiovisual programming in ChucK.” *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*, Utrecht, The Netherlands.

Alain, G., Chevalier-Boisvert, M., Osterrath, F., & Piché-Taillefer, R. (2020). DeepDrummer: Generating drum loops using deep learning and a human in the loop. *Proceedings of the Joint Conference on AI Music Creativity (CSMC + MuMe)*.

Assayag, G., Bloch, G., Chemillier, M., Cont, A., & Dubnov, S. (2006). “OMax Brothers: A dynamic topology of agents for improvisation learning.” *Proceedings of the ACM Workshop on Audio and Music Computing Multimedia* (pp. 125–132).

Brosnan, T. (2026). “DARC: Drum accompaniment generation with fine-grained rhythm control.” arXiv:2601.02357.

Bruford, F., McDonald, S., & Sandler, M. (2020). “jaki: User-controllable generation of drum patterns using an LSTM encoder-decoder and deep reinforcement learning.” *Proceedings of the Joint Conference on AI Music Creativity (CSMC + MuMe)*.

Burloiu, G. (2020). “Rolypoly~: Learning expressive timing and dynamics in drum patterns.” *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Late Breaking Demo.

Callender, L., Hawthorne, C., & Engel, J. (2020). “Improving perceptual quality of drum transcription with the Expanded Groove MIDI Dataset.” arXiv:2004.00188.

Cover, T. M., & Hart, P. E. (1967). “Nearest neighbor pattern classification.” *IEEE Transactions on Information Theory*, 13(1), 21–27.

Delgado, A., McDonald, S., Xu, N., & Sandler, M. B. (2019). “A new dataset for amateur vocal percussion analysis.” *Audio Mostly 2019: A Journey in Sound* (pp. 1–7). https://doi.org/10.1145/3356590.3356844

Evans, N., Haki, B., & Jordà, S. (2024). “GrooveTransformer: A generative drum sequencer Eurorack module.” *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*, Utrecht, The Netherlands.

Gillick, J., Roberts, A., Engel, J., Eck, D., & Bamman, D. (2019). “Learning to groove with inverse sequence transformations.” *International Conference on Machine Learning* (pp. 2269–2279). PMLR.

Haki, B., Nieto, M., Pelinski, T., & Jordà, S. (2022). “Real-time drum accompaniment using transformer architecture.” *Proceedings of the 3rd Conference on AI Music Creativity (AIMC 2022)*, Saint-Étienne, France.

Li, Y., & Wang, G. (2024). “ChAI: Interactive AI tools in ChucK.” *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME),* Utrecht, The Netherlands.

Madaghiele, V., & Cotton, K. (2025). “Recursive Speculation.” *Proceedings of the 6th Conference on AI Music Creativity (AIMC 2025)*, Brussels, Belgium.

Madaghiele, V., Fasciani, S., Kelkar, T., & Erdem, Ç. (2025). “MAAL: A multi-agent autonomous live looper for improvised co-creation of musical structures.” *Proceedings of the 6th Conference on AI Music Creativity (AIMC 2025),* Brussels, Belgium.

Marchini, M., Pachet, F., & Carré, B. (2017). “Rethinking reflexive looper for structured pop music.” *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)* (pp. 139–144). Aalborg University Copenhagen, Denmark.

Martanto, J., & Kartowisastro, I. H. (2025). “Beatbox classification to distinguish user experiences using machine learning approaches.” *Journal of Computer Science,* 21(4), 961–970.

Nuttall, T., Haki, B., & Jordà, S. (2021). “Transformer neural networks for automated rhythm generation.” *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*, Shanghai, China.

Pachet, F. (2003). “The Continuator: Musical interaction with style.” *Journal of New Music Research,* 32(3), 333–341.

Ramires, A., Penha, R., & Davies, M. E. P. (2018). “User specific adaptation in automatic transcription of vocalised percussion.” arXiv:1811.02406.

Santos, A., Cardoso, A., Davies, M. E. P., & Dannenberg, R. B. (2025). “A synthetic strategy for automatic drum TapScription.” *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*.

Shepardson, V., & Magnusson, T. (2023). “The Living Looper: Rethinking the musical loop as a machine action-perception loop.” *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)* (pp. 1–8). Mexico City, Mexico.

Shepardson, V., Stefánsdóttir, H. S., & Magnusson, T. (2025). “Evolving the Living Looper.” *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*.

Stowell, D., & Plumbley, M. D. (2010). “Delayed decision-making in real-time beatbox percussion classification.” *Journal of New Music Research,* 39(3), 203–213.

Wang, G., Cook, P. R., & Salazar, S. (2015). “ChucK: A strongly timed computer music language.” *Computer Music Journal,* 39(4), 10–29.

Weber, P., Uhle, C., & Müller, M. (2024). “Real-time automatic drum transcription using dynamic few-shot learning.” *Internet of Sounds.* Fraunhofer IIS.