Scaffolding Off-the-Shelf Text-to-Audio Models for Real-Time Generation in Live Performance

Louis McCallum*
Creative Computing Institute
University of the Arts, London
London, UK
l.mccallum@arts.ac.uk

Anna Wszeborowska
Creative Computing Institute
University of the Arts, London
London, UK
a.wszeborowska@arts.ac.uk

Abstract

Contemporary text-to-audio generation models face two independent technical challenges when deployed in live performance: buffer-quantised input control and subreal-time generation speed. Live musicians and technologists often work with off-the-shelf models and cannot address these issues at their source, requiring external scaffolding solutions. We present a taxonomy of architectural strategies to address these constraints. We analyse tradeoffs in latency, continuity, and computational cost, discussing practical implications for system design. A case study with Meta's MusicGen (Copet et al., 2023) text-to-audio model demonstrates these strategies in practice. We characterise each strategy along two independent axes: the time the user spends waiting for requests to take effect (stale time), and the proportion of audio that is unique rather than recycled (novelty). Pre-computation through crossfading reduces median input-to-playback latency from 5.34 s (looping baseline) to 0.03 s and eliminates stale time entirely, at the cost of low novelty; parameter-space interpolation achieves both responsiveness and novelty but requires the largest setup investment.

1 INTRODUCTION

Machine learning powered music generation has achieved significant quality improvements in recent years (Hantrakul et al., 2019; Caspe et al., 2025; Caillon and Esling, 2022), yet deployment in live performance contexts remains challenging for models not explicitly designed for this. Two distinct technical problems constrain real-time use, though they are often conflated in practice. This conflation obscures the fact that different strategies are required depending on which constraint exists, and that some approaches can address both simultaneously.

1.1 Buffer-Quantised Input

Most neural audio models generate audio in fixed-length buffers ranging from 2048 samples to several seconds rather than producing continuous streams. This is especially prevalent in text-to-audio models. This architectural constraint means that performer input can only affect generation at buffer boundaries, limiting control bandwidth far below the 30–50 Hz typical of acoustic instruments. Although recent work has optimised neural audio synthesis architectures for low latency through approaches like BRAVE (Caspe et al., 2025) and streaming RAVE (Caillon and Esling, 2022), the majority of models available to practitioners are pre-trained, non-streaming models. The reasons for this reliance are varied: streaming-trained models remain scarce, retraining requires expertise and computational resources that many practitioners lack, particular pre-trained models may exhibit desirable characteristics, and practitioners need techniques applicable across diverse generative architectures. We therefore focus on off-the-shelf, pre-trained models that musicians and technologists can use immediately without modification, acknowledging that purpose-built low-latency models exist but seeking strategies that enable existing models to function in performance contexts.

1.2 Subreal-Time Generation

Computationally expensive models may be unable to generate audio faster than playback speed, and even models that achieve adequate throughput on high-specification hardware may not perform similarly on equipment available to performers. A model requiring 3 s to generate 2 s of audio operates at $0.67\times$ real time and cannot sustain continuous playback without precomputation. Echo Chamber (Lim et al., 2024), an installation using MusicGen (Copet et al., 2023) to generate continuous audio experiences for its participants, uses looping and layering to compensate for subreal generation times. Even models achieving average real-time ratios above unity may experience occasional slowdowns that create unpredictable gaps in output. ReaLJam (Scarlatos et al., 2025), a system for live jamming that uses large Transformer models, implements a fallback mechanism that ensures uninterrupted playback when a response takes longer than one frame by playing cached chords previously scheduled for upcoming frames.

1.3 The Off-the-Shelf Constraint

The increasing availability of audio models in accessible wrappers represents a positive development for enabling their use by a broad range of musicians. However, users remain limited in how they can address the constraints outlined above. Often unable to reconfigure model architecture due to lack of expertise or to retrain due to lack of computational resources, practitioners require external scaffolding strategies that work within model limitations rather than modifying the models themselves.

Often models are behind paywalls or subscription APIs, or only results (not weights) are released for academic publication. This restricts certain state-of-the-art work (in terms of generation times) for general use by musicians. We

*Code available at https://github.com/louismac/scaffolding-text-2-audio

focus on available models as they must be on-the-shelf to be taken off-the-shelf. We are also motivated by a realistic constraint of availability. Although some models represent impressive generation times on high-powered hardware, this is often not financially or logistically feasible (e.g., "gig-ready") for musicians.

1.4 Scope and Contributions

This work focuses on architectural strategies deployable with off-the-shelf text-to-audio models where the underlying generation algorithm cannot be modified, a common constraint in production environments. Our approach is designed to be generalisable and to some extent future-proof as new models emerge. The contributions of this paper are: (i) a formal problem decomposition that separates buffer quantisation from generation speed, two constraints commonly conflated in the literature; (ii) a unified taxonomy that places several previously separate techniques (loop-based generation, overlapping streaming, parameter-state crossfading, trajectory pregeneration, parameter-space interpolation, and granular-hybrid synthesis) within a single framework organised by which constraint each addresses; (iii) analysis of tradeoffs in control latency, audio continuity, and computational cost across the taxonomy; (iv) practical recommendations for strategy selection based on model characteristics and performance requirements; and (v) a case study with the MusicGen (Copet et al., 2023) text-to-audio model that operationalises each strategy and reports comparative latency, stale-time, and novelty measurements. We do not claim novelty for the individual scaffolding techniques themselves — many derive from established traditions in granular and concatenative synthesis — but rather for their unified treatment as scaffolding around modern neural audio models, and for the empirical comparison.

2 RELATED WORK

2.1 Real Time Audio Systems

2.1.1 Latency in Digital Signal Processing (DSP)

Anderson and Kuivila (1986) established foundational principles for real-time musical computation, demonstrating that interactive systems require predictable timing within perceptual thresholds. Traditional DSP-based systems achieve 1.5–11.6 ms latency with 64–512 sample buffers at 44.1 kHz (Anderson and Kuivila, 1986). Neural generation often violates these assumptions as generation time is variable, buffer requirements can be orders of magnitude larger, and completion within buffer duration cannot be guaranteed (Caspe et al., 2025).

Recent work on neural synthesiser latency (Caspe et al., 2025) shows that even optimised architectures struggle to achieve sub-20 ms response suitable for expressive control. The authors demonstrate that compression ratio directly impacts both buffering latency and temporal jitter, creating fundamental tradeoffs in model design that cannot be circumvented through external scaffolding alone.

2.1.2 Latency Perception

Human perception studies establish that latencies above 10–20 ms become noticeable, with degradation in performance quality beyond 30–50 ms (Chew et al., 2004; Rottondi et al., 2016). Networked music performance research identifies 20–30 ms as the critical threshold for maintaining synchronous interaction (Rottondi et al., 2016). Recent investigations reveal nonlinear perception thresholds as musicians show mean just-noticeable differences of 49 ms for 0 ms base latency but only 27 ms for 64 ms base latency (Schmid et al., 2024), suggesting context-dependent adaptation that may inform how performers adjust to buffer-scale latencies.

Jack et al. (2016) demonstrate that even moderate latencies affect both objective performance metrics and subjective quality assessments in digital musical instruments. This finding confirms that the buffer-scale latencies inherent in many off-the-shelf models are fundamentally incompatible with traditional interactive performance paradigms, motivating the exploration of alternative interaction models.

2.2 Manipulating Buffers in Performance

2.2.1 Audio Resynthesis

Granular synthesis (Roads, 1988; Truax, 1988) achieves responsiveness through small grain sizes of 10–100 ms and precomputed material. Truax's real-time implementation on dedicated signal processors (Truax, 1988) demonstrates that millisecond-scale segments enable interactive control while maintaining timbral coherence. These techniques provide templates for hybrid approaches that combine neural generation with traditional synthesis methods.

Concatenative synthesis extends content-based segment selection to real-time contexts. Schwarz's CataRT (Schwarz et al., 2006) uses descriptor-based corpus navigation, while recent work (Tralie and Cantil, 2024) achieves Bayesian-guided selection with computational complexity independent of corpus size. However, these systems operate at millisecond timescales, not the multisecond buffers required by many neural models, necessitating adaptation for the neural audio context.

2.2.2 Audio Morphing

Spectral morphing techniques (Westerfeld, 2018) enable real-time interpolation between timbres through additive resynthesis. Recent work (Kamath et al., 2025) explores semantic interpolation in learned latent spaces, suggesting possibilities for neural model control through pregeneration and real-time blending that inform several strategies in our taxonomy.

2.2.3 Streaming Neural Audio

Caillon and Esling (2022) developed post-training methods for converting noncausal convolutional models to streaming architectures. Their work on RAVE demonstrates that cached-padding mechanisms enable buffer-based processing without quality degradation, though minimum buffer sizes remain determined by compression ratios and typically cannot fall below 2048 samples. This results in a latency of at least 46.4 ms at a 44.1 kHz sampling rate.

2.2.4 Mitigating Latency in Generation

Some existing systems implicitly adopt strategies in this taxonomy. Riffusion's interactive demo, for instance, achieves responsiveness through latent-space interpolation between pre-generated spectrogram chunks, an instance of parameter-space interpolation (Forsgren and Martiros, 2022). Magenta RealTime (Caillon et al., 2025) uses chunk-based autoregression with coarse representation of audio context in order to achieve real-time generation of

infinite and uninterrupted audio streams in live generation settings.

3 PROBLEM FORMALISATION

3.1 Buffer-Quantised Input

For a system generating audio in fixed chunks (buffers), the control update frequency f_c - which denotes how often the system can react to user inputs per second - is given by

$$f_c = \frac{1}{B}, \tag{1}$$

where B represents the generated buffer length in seconds. For B ranging from 100 ms to 4 s, f_c ranges from 10 Hz down to 0.25 Hz, compared with 30–50 Hz for acoustic instruments. Control latency L_c denotes the lag between a user input and its effect in the generated output. In the absence of scaffolding, L_c satisfies

$$L_c \geq \max(B, T_g), \tag{2}$$

where T_g is the time to generate a buffer (defined in the following subsection). When generation is faster than playback ($T_g \leq B$), the bound reduces to $L_c \geq B$ and is set by buffer quantisation; when generation is slower ($T_g > B$), the bound is set by generation time itself. In either case the bound exceeds established interactive thresholds of 20–50 ms by one to two orders of magnitude.

3.2 Subreal-Time Generation

Let T_g denote the generation time for a buffer length B. The generation ratio is defined as

$$R_g = \frac{B}{T_g}. \tag{3}$$

Continuous playback requires $R_g \geq 1.0$, meaning the duration of the generated audio buffer (B) must be at least as long as the time taken to compute it (T_g). Models with $R_g < 1.0$ cannot sustain real-time playback without precomputation. Variable generation times create uncertainty even when average R_g exceeds unity, as occasional slowdowns interrupt output.

3.3 Problem Independence

These two constraints are logically independent. A model with $R_g = 10.0$, meaning very fast generation, still exhibits $f_c = 0.5$ Hz if its buffer size $B = 2$ s. Conversely, a model with theoretical $f_c = 100$ Hz but $R_g = 0.5$ cannot maintain continuous playback regardless of its fine temporal resolution. Most practical models exhibit both $R_g < 1.0$ and low f_c simultaneously. Effective solutions must identify and address the binding constraint or constraints for specific deployment contexts, and some strategies can address both problems through different mechanisms.

4 STRATEGY TAXONOMY

The strategies presented here either enable performance when generation is too slow ($R_g < 1.0$), improve reaction times (f_c) when buffers are long (high B), or address both constraints through hybrid approaches.

4.1 Loop-Based Generation

Loop-based generation produces material designed for seamless looping, repeating the current buffer while generating the next variation. Implementation can avoid artefacts

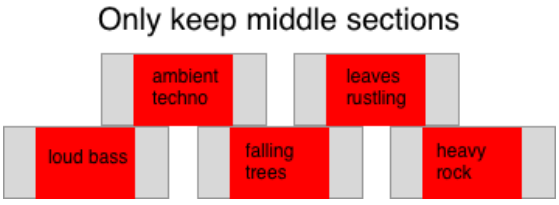

Figure 1: Overlap Streaming

by ensuring amplitude and phase continuity at buffer endpoints. For $R_g = 0.5$, a minimum of two repetitions is required before a new buffer becomes available.

This approach tolerates arbitrary R_g values through repetition. Control latency is variable, determined by the number of repetitions multiplied by B, but performers control transition timing and can treat this variability as a compositional parameter. Computational cost is low since we are repeating existing generated material.

4.2 Overlapping Buffer Streaming

In situations where minimum buffer length B is architecturally constrained, overlapping buffer streaming generates buffers with temporal overlap, retaining only well-contextualised centre portions while discarding edges. The implementation uses a stride parameter $s = B/S$, where S (the overlap factor) typically ranges from 2 to 6. For $S = 6$ and $B = 2$ s, the stride becomes approximately 333 ms. The system retains only the centre region of each buffer where the model has full context, then crossfades between adjacent centres.

This reduces effective control latency to the shorter stride duration $L_c = s$ rather than the full buffer length B, though it require a generation ratio of $R_g \geq S$ to maintain continuous generation. The approach can achieve 6× latency improvement, bringing control latency into the 200–500 ms range with good continuity through smooth crossfades. However, it worsens the generation speed constraint by requiring generation speed at least S times faster than playback. Consequently, the computational cost is high since the system generates S times more audio than is ultimately used.

4.3 Crossfading Between Parameter States

This strategy maintains multiple parallel generation streams with different parameter configurations, crossfading between streams when parameters change. Implementation maintains N_{streams} concurrent processes. On parameter change, the system spawns a new stream with target parameters, crossfades over transition duration T_{fade} typically ranging from 100–500 ms, then discards the old stream. Effective latency becomes $L_c = B + T_{\text{fade}}$, requiring $R_g \geq N_{\text{streams}}$ to sustain all parallel processes.

The approach reduces latency from $N \cdot B$ to fade time. Continuity can be good with proper crossfading, though standard amplitude crossfading risks phase cancellation between incoherent sources. Computational cost is very high due to multiple simultaneous streams requiring independent generation processes.

4.4 Trajectory Pregeneration

When computational constraints prevent $R_g \geq 1.0$, pregeneration can provide an approach that constrains possibility

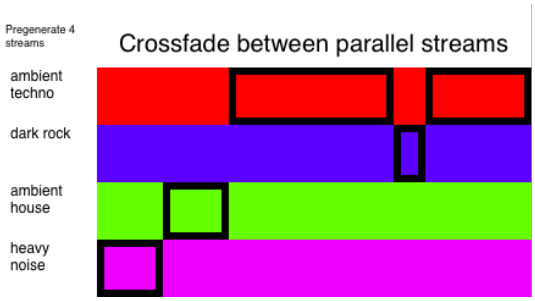

Figure 2: Crossfade between parallel streams

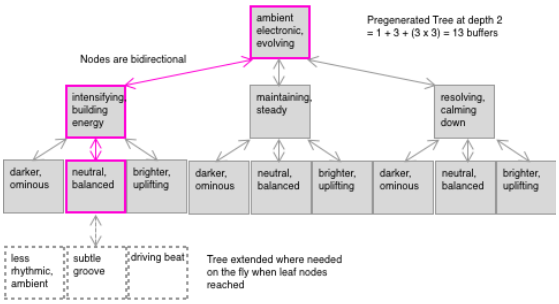

Figure 3: Navigate a pregenerated tree

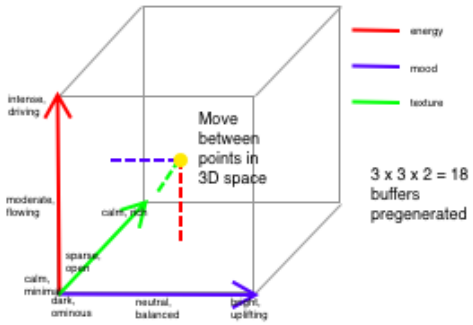

Figure 4: Move through points in 3D space

space while preserving compositional choice during performance. The design challenge becomes determining what to generate and how to present transitions to performers in a musically meaningful way.

One implementation constructs a decision tree by pregenerating multiple continuations from each buffer. For each buffer, the system computes M branches with different parameter trajectories and maintains tree depth D. Tree size grows as M^D, constraining practical depth. When the performer chooses a branch, the system retains that subtree and generates new branches at the frontier as computational resources permit.

The entire tree can be generated offline or asynchronously, tolerating arbitrary R_g values. However, this strategy does not address buffer quantisation, maintaining $L_c = B$ at decision points. Control latency equals one buffer length with excellent continuity since all branches are designed for coherent continuation. Computational cost at initialisation is very high due to exponential growth with tree depth.

4.5 Parameter Space Interpolation

Parameter space interpolation pregenerates buffers at parameter space extremes offline, then uses traditional synthesis techniques for real-time interpolation between cached buffers. The synthesis proceeds in three phases. During offline preparation, the system generates buffers at grid points in parameter space, for example an 8×8 grid in two-dimensional space yielding 64 cached buffers. During real-time performance, the system uses wavetable interpolation, spectral morphing (Westerfeld, 2018), or granular techniques to blend between cached buffers based on continuous parameter input. The cache is periodically regenerated when performance context changes.

This approach addresses buffer quantisation by enabling audio-rate control with update frequencies in the kilohertz range, limited only by the interpolation algorithm rather than buffer boundaries. It simultaneously addresses subreal-time generation since offline computation tolerates arbitrary R_g values. Control latency becomes submillisecond for the interpolation itself, though cache regeneration incurs high latency when needed. Continuity depends on the chosen interpolation technique and the smoothness of the underlying parameter space.

4.6 Granular-Hybrid Synthesis

Granular-hybrid synthesis uses the generative model to create short fragments of 200–1000 ms, then applies granular or concatenative synthesis techniques for real-time arrangement. During offline or background processing, the system generates a library of grains with varying parameters. During real-time performance, it applies grain selection, playback rate modulation, and spatial positioning based on performer input. The system can periodically update the library with newly generated material during performance gaps.

This strategy addresses buffer quantisation through grain-level control operating at 10–100 ms timescales, well below typical buffer durations. It addresses subreal-time generation by decoupling library generation from playback, tolerating arbitrary R_g values. Control latency remains low at 10–100 ms for grain manipulation, though library updates incur higher latency. Continuity varies depending on grain overlap strategy and selection algorithm, with the inherently fragmented texture of granular synthesis potentially masking discontinuities.

5 CASE STUDY: TEXT-TO-AUDIO WITH MUSICGEN

MusicGen is a text-to-audio model from Meta available in small, medium, and large variants, each occupying different positions on the generation speed versus output quality tradeoff spectrum. As a widely available off-the-shelf model, it provides an appropriate testbed for evaluating the strategies presented above.

5.1 MusicGen Buffer Size Limitations

MusicGen generates audio at 50 tokens per second. Generation time can be modeled as

$$T_{\text{total}} = T_{\text{startup}} + (n_{\text{tokens}} \times T_{\text{per-token}}) \quad (4)$$

where T_{startup} encompasses text encoding, key-value cache initialisation, and first-token generation. For small buffers, this fixed cost dominates and significantly reduces the real-time ratio R_g.

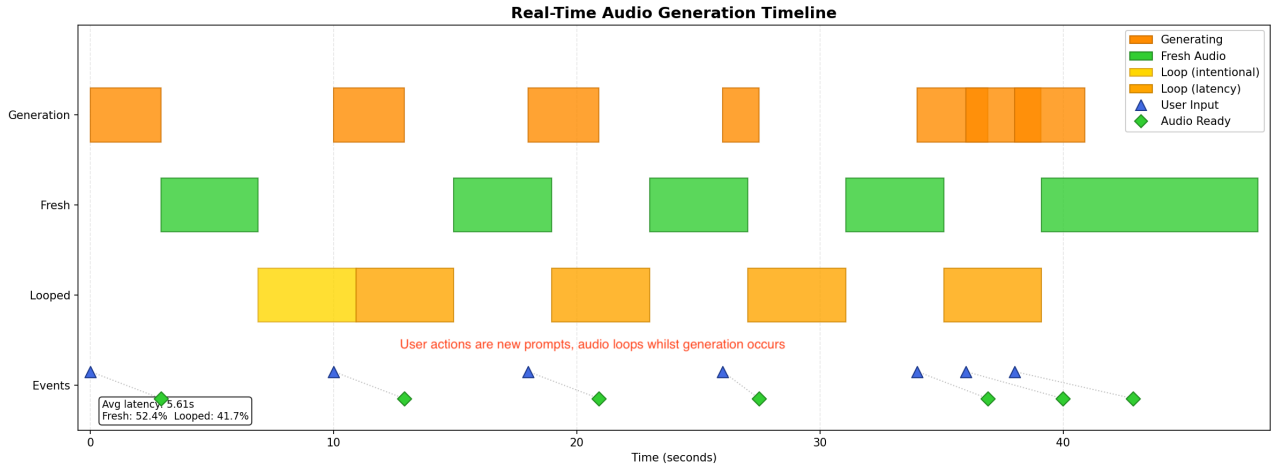

Figure 5: Plot of performance using the Looping approach, demonstrating responsiveness and freshness to typical performer interactions.

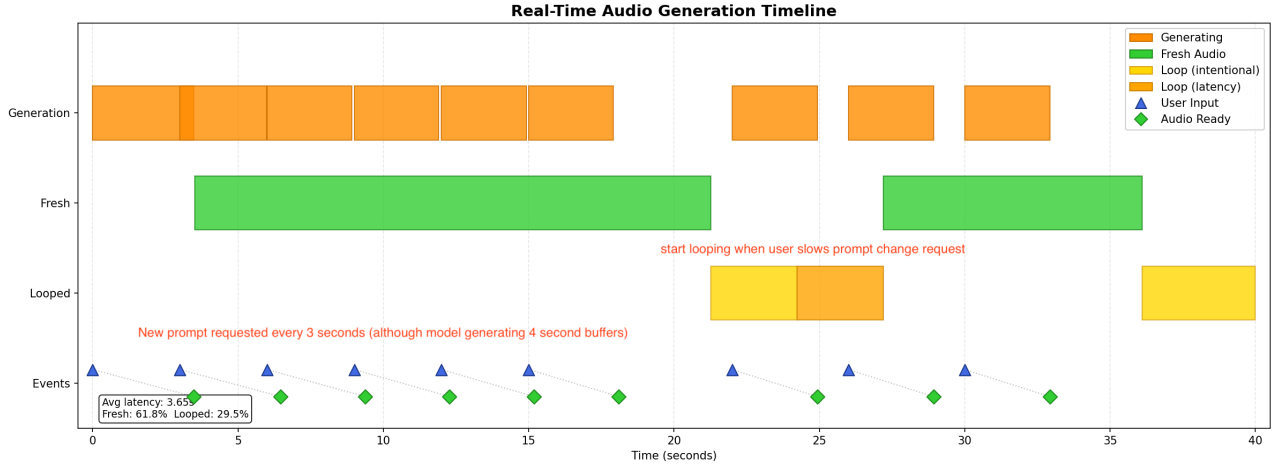

Figure 6: Plot of performance using the Overlap approach. Overlapping buffers allows for new prompts to enter the performance quicker (lower interaction latency).

The model requires sufficient context to generate coherent music. At 25 tokens representing 0.5 s, approximately one to two beats at 120 BPM are represented, which proves insufficient for the model to establish rhythm, melody, or harmonic structure. MusicGen employs EnCodec at 24 kHz with 4 codebooks for audio encoding, and the decoder requires a minimum number of frames for proper reconstruction. Very short sequences produce artefacts at boundaries that degrade output quality.

Transformer attention patterns produce lower-quality output at sequence boundaries. A common mitigation strategy retains only the centre portion of generated audio, but with small buffers this approach discards the majority of generated content and becomes impractical.

A fundamental tension therefore exists between control latency and output quality. Smaller buffers enable faster response to prompt changes but yield lower quality and reduced computational efficiency. Larger buffers produce higher quality output with greater efficiency but result in sluggish response to user input. This tradeoff motivates the mitigation strategies presented, which represent workarounds for this fundamental architectural constraint.

5.2 Method

As a text-to-audio model, MusicGen provides distinctive affordances for interaction and control compared to other audio generation models that may rely on continuations, timbre transfer, or latent space interpolations. For each approach formalised above, we present implementations using MusicGen where the primary form of interaction is the current prompt used to generate audio. The performer may select a completely new prompt or add decoration to the current prompt for more subtle change. They can also affect the guidance level, controlling how strongly the prompt conditions the output, and the temperature of generation. Any change to these parameters requires a new buffer to be requested and generated.

For each approach, we measure buffer request timing, buffer delivery, user action, and received outcome. This demonstrates the practical effects of each solution on the two key problems we seek to address. We also provide commentary on the aesthetic implications that accompany each choice, as strategy selection influences not only technical performance but also the character of the resulting musical output.

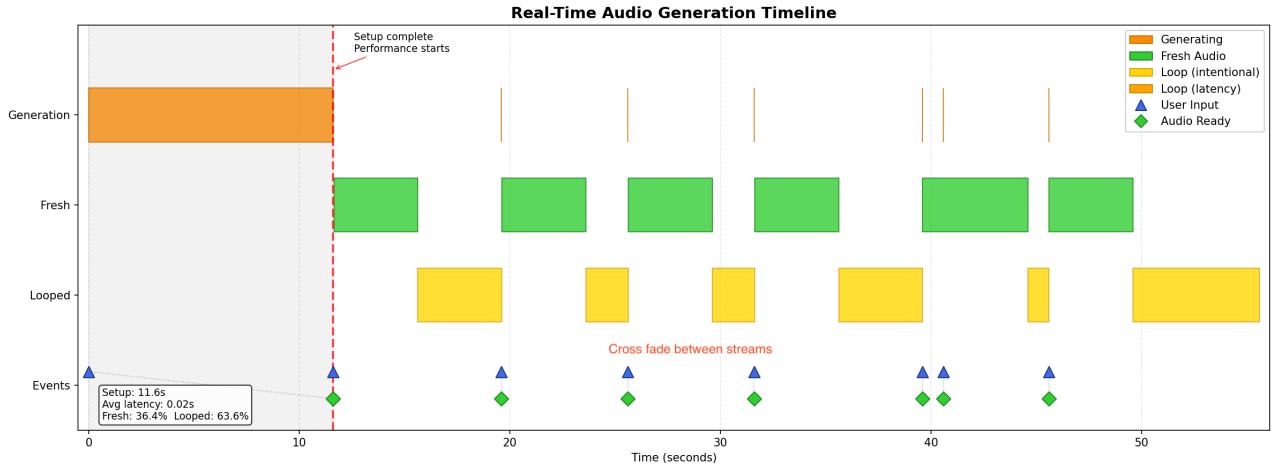

Figure 7: Plot of performance using the Crossfade approach. After some initial setup delay, response to performers interactions is near immediate.

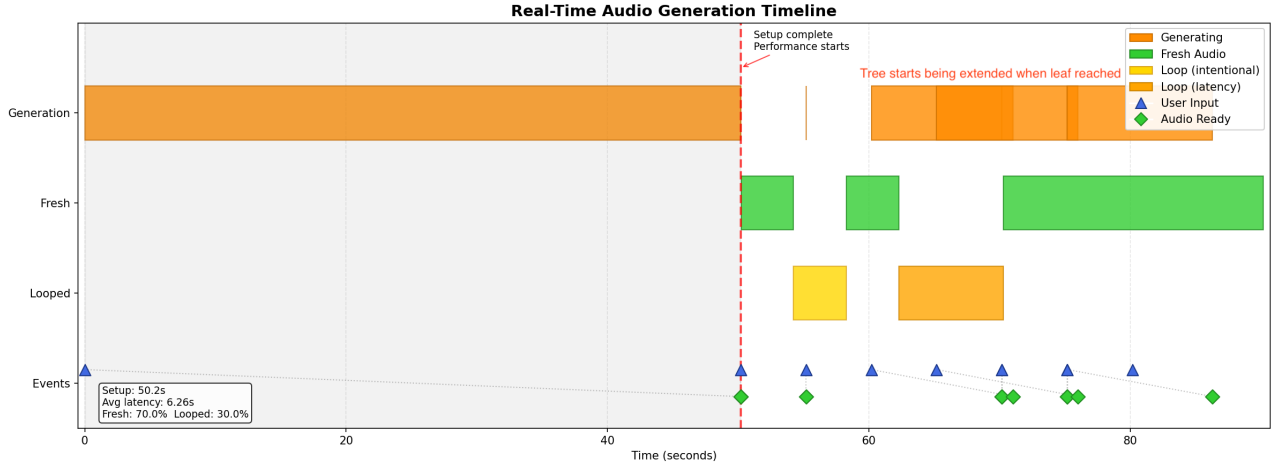

Figure 8: Plot of performance using the Trajectory Pregeneration approach. When using the initially generated tree, model response is quick, however, once we have reached the maximum depth this slows whilst the tree is extended.

5.2.1 Looping Implementation

This approach starts with one user-inputted prompt and keeps looping that audio until the user changes the prompt and new audio becomes available, at which point the system fades into the new audio at the next loop point. This has minimal startup costs but f_c is limited by T_g and B. This represents the minimal behaviour across all strategies, as the current buffer will keep playing until the user has interacted and the interaction result is ready to be rendered.

5.2.2 Overlapping Buffer Streaming Implementation

Shown in Fig. 1, this approach relies on small R_g. Given sufficient generation speed, we can generate buffers on the fly that contain new prompts with high f_c because only the central part is kept and crossfaded with new audio. This method demonstrates that without sophisticated pregeneration strategies, f_c is bounded only by R_g when generation is sufficiently fast.

5.2.3 Crossfading Implementation

Heavy on pregeneration startup costs, this approach enables very high f_c as we generate multiple streams with multiple prompts and crossfade between them rapidly when desired (see Fig. 2. The precomputation investment pays dividends in responsiveness during performance.

5.2.4 Trajectory Pregeneration Implementation

We create a tree at initialisation consisting of an original prompt and branches that allow for chosen modifiers of aspects such as energy, texture, mood, rhythm, and development (see Fig. 3). The children are generated with the base prompt plus the modifier. Each depth level is a new modifier, with the children gaining more descriptive aspects to the prompt as we move down. The tree is bidirectional so we can go back to parent nodes. Once we reach a leaf node, the children are generated on the fly to keep extending the tree (depth is limited depending on how many modifier options are available). There are substantial startup costs, but depending on tree depth and user desire to switch to a new base prompt, controllability is not constrained by T_g. The approach requires preplanning of both base prompts and logistics to trigger pregeneration before performance can begin.

5.2.5 Parameter Space Interpolation Implementation

Shown in Fig. 4, we define a grid across three categories covering mood, energy, and texture, with each category

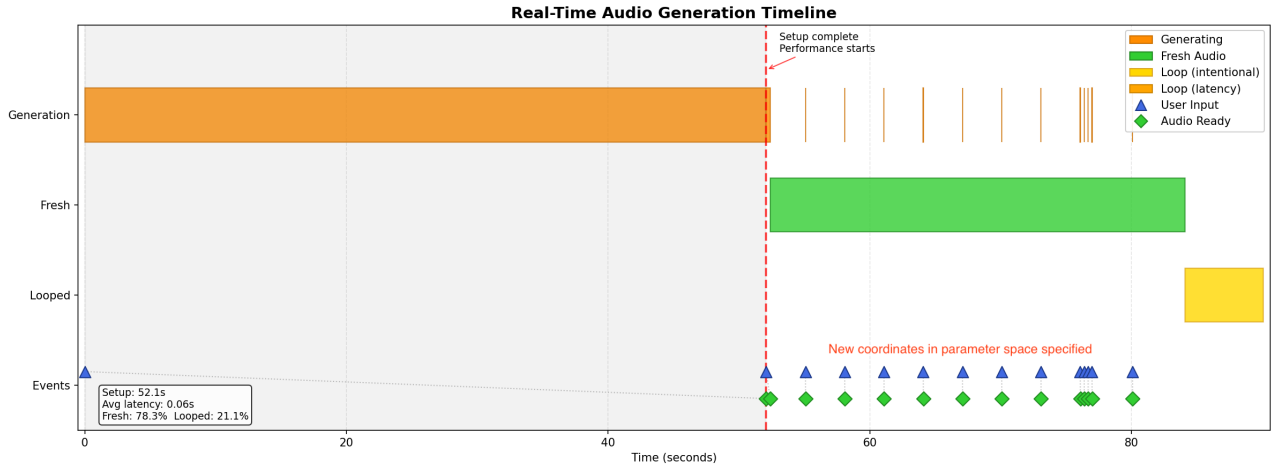

Figure 9: Plot of performance using the Parameter Space Interpolation approach. Long setup allows for fast and continuous performer control.

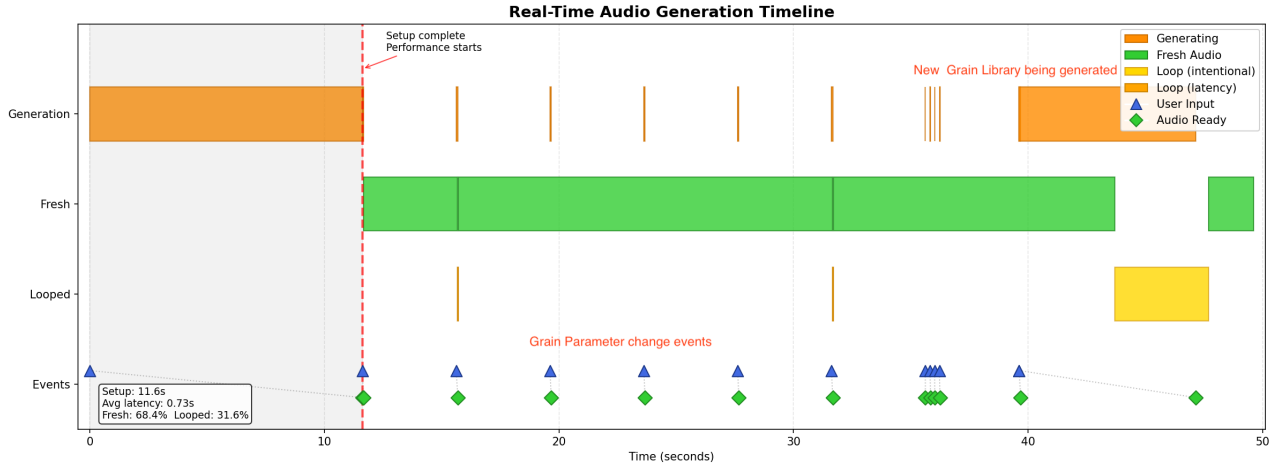

Figure 10: Plot of performance using the Granular Synthesis approach. Here novel audio is measured slightly differently as playback can contain high variability and responsiveness whilst using the same source buffer.

containing three to four states. We then define points on the resulting hypercube and crossfade between them, achieving high f_c from high pregeneration costs. As with other pregeneration methods, interactivity is bounded by what has been pregenerated, but we gain expressive possibilities through continuous movement in high-dimensional parameter space rather than discrete jumps.

5.2.6 Granular Synthesis Implementation

This approach builds a grain library from a set of prompts, with each grain representing a variation produced by adding noise to the guidance factor. Here we sacrifice long-term coherence of sample buffers for much greater control that is also highly responsive. The parameters of granular synthesis including pitch shift, envelope, position, spread, width, and density are fine-grained and controllable in near real-time. Rebuilding the library with a new prompt takes time but is possible, and meaningful interaction can continue while regeneration occurs in the background.

5.3 Results

For each strategy, we implemented it in Python and programmed a fixed sequence of representative performer interactions, including prompt changes, guidance-level adjustments, and idle periods. Whilst we try to maintain comparability between simulations as much as possible, as each strategy represents differing control affordances, the sequences themselves differ in some places.

We then used a real-time simulation harness to play back the sequenced performances and track generation times, latency between user actions and audible response, and the proportion of forced versus intentional looping of content. Reported figures reflect a single representative run per condition; we discuss variance and statistical reporting as a limitation in Section 7. Tables 1 and 2 report results for the MusicGen Small and Medium variants respectively on an RTX 2080 generating 4-second buffers. On a lower-powered machine, or using a larger model, generation times are longer and this affects setup times and looping latency depending on approach. For space reasons we include timeline plots only for representative configurations, but the qualitative tradeoffs generalise.

The Looping performance (Fig. 5) demonstrates that depending on how close to the end of a buffer the user action to change prompt comes, a loop occurring whilst waiting is more or less likely. If multiple requests come in before the first one has completed generating (as in towards the end), then we can get an extended section of novel audio

Table 1: Cross-strategy performance comparison for real-time audio generation, MusicGen Small on RTX 2080 GPU with $B = 4$ s. All values are aggregated over $n = 10$ independent runs per strategy with identical interaction sequences. Setup and Avg Latency are reported as median (IQR); Min and Max Latency are the across-run minimum and maximum observed within each strategy. Latency is the elapsed time from a user prompt change to the first sample of audio reflecting that change. Stale is the percentage of playback time spent repeating audio while the user waits for new generation to complete; lower is better, with 0 indicating the user never has to wait. Novel is the percentage of playback time during which the audio is unique rather than repeated content; higher indicates greater content variety

Strategy	Setup	Input-to-playback latency (s)			Stale	Novel
	(s)	Avg	Min	Max	(%)	(%)
Loop	0.00	5.34	2.96	8.70	32.7	48.9
Overlap	0.00	5.03	2.92	6.48	22.2	63.1
Crossfade	11.67 (0.71)	0.03 (0.01)	0.00	0.05	0.0	19.4
Trajectory	47.15 (0.09)	9.47 (0.00)	0.05	23.75	24.7	63.0
Param Space	52.46 (0.07)	0.03 (0.00)	0.01	0.06	0.0	79.0
Granular	11.68 (0.22)	0.85 (0.00)	0.01	8.69	21.1	57.9

Table 2: Cross-strategy performance comparison, MusicGen Medium on RTX 2080 with $B = 4$ s, aggregated over $n = 10$ independent runs per strategy with the same interaction sequence as Table 1. Stale and Novel are defined in Table 1.

Strategy	Setup	Input-to-playback latency (s)			Stale	Novel
	(s)	Avg	Min	Max	(%)	(%)
Loop	0.00	8.18	4.87	10.21	45.0	34.3
Overlap	0.00	25.10 (3.74)	13.62	35.70	36.9	6.7
Crossfade	26.51 (0.50)	0.03 (0.01)	0.00	0.05	0.0	19.4
Trajectory	144.99 (1.73)	6.47 (0.01)	0.05	14.71	24.7	63.0
Param Space	123.22 (0.53)	0.03 (0.00)	0.01	0.06	0.0	79.0
Granular	26.65 (0.33)	0.06 (0.00)	0.01	0.09	21.1	47.4

as when the end of a buffer is reached, the most recently generated audio is played.

The Overlap performance (Fig. 6) shows that on this particular combination of model and machine (generating 4s buffers with the centre 3s kept), we are able to keep novel audio continuously running. We request new prompts every 3 seconds in the early stages of the performance and the model is able to deliver on this because we have a low R_g; on other configurations this approach would not have been possible. For the medium sized model, the Overlap is never able to generate audio in time to keep up with the sub-B prompt updates so is perpetually and increasingly building up a latency debt.

Crossfade shows a reasonable set up cost, but then almost immediate response to user input (seen by alignment of triangle and diamond in Fig. 7). The Trajectory Pregeneration for a depth of 2 has nearly a minute set up costs (so this would also be the cost to regenerate the tree mid performance) but then again allows for near immediate response to user input (Fig. 8). Although the tree is bidirectional so all buffers could be used, there is the potential for wasted generation here with buffers generated and not played. We see a long latency whilst we wait for new depth to be added to the tree once a leaf node has been reached.

Parameter Space Interpolation (Fig. 9) and Granular Synthesis (Fig. 10) also trade startup costs for controllability and responsiveness (although at a lower startup cost for the latter). We see the generation time of rebuilding a new library at the end of this plot. The Novel metric for Granular Synthesis reflects the proportion of playback during which grains derive from the current parameter state rather than a stale library snapshot, since the library is regenerated only when prompts change.

5.4 Discussion

The strategies cluster into two distinct groups based on their preparation requirements. Zero-setup approaches including loop and overlap require no preparation time but exhibit higher or more inconsistent latency. Pre-computed approaches including Trajectory, Crossfade, Parameter Space, and Granular strategies require 11–52 s of setup but enable near-instant response, with median latencies of 0.03–0.85 s.

Stale is a property of the strategy itself as it characterises whether the system is structurally capable of avoiding user waits. Novel is a property of the strategy combined with the performance. Crossfade's 19.4% novelty in this evaluation reflects a performance that revisits its small set of pre-generated streams and a performance that triggered cache regeneration more often, or a Crossfade variant configured with more streams, would yield higher novelty without changing the Stale figure. Likewise, Parameter Space's 79.0% novelty assumes the performer explores the grid. A performer fixated on one corner would see lower novelty despite the strategy's full capacity for it. Stale therefore captures what a strategy guarantees and Novel captures what a strategy enables given performer demand. Absolute Novel values reported here relate to our specific simulated interaction sequence but are demonstrative of typical performance affordances between strategies.

The Crossfade strategy achieves zero stale time at modest setup cost (median 11.7 s), but pays for this with low novelty (19.4%) a the system switches instantly between a small set of pre-generated streams, but those streams themselves are pre-baked and cycle. This is the right tradeoff when responsiveness matters more than content variety, for instance when the performer wants tight coupling between

gesture and timbral change but is content for the underlying material to repeat.

Parameter Space Interpolation occupies an arguably more attractive corner of the design space, achieving both zero stale time and the highest novelty in our comparison (79.0%). The cost is the highest setup time of any strategy (median 52.5 s) and the constraint that exploration is bounded by the pre-generated grid. For performance contexts where setup time is acceptable and the parameter space is known in advance, this strategy dominates.

Granular synthesis offers a middle ground: 0% stale time while continuously generating new grains gives 57.9% novelty, with low setup cost (11.7 s) comparable to crossfade. The tradeoff appears in average latency (0.85 s, with worst-case 8.69 s), reflecting library regeneration during performance. This makes it well suited to textural rather than transient-driven performance.

The Loop, Overlap, and Trajectory strategies all exhibit substantial stale time (32.7%, 22.2%, and 24.7% respectively). Loop and Overlap have zero setup cost but incur this stale time on essentially every prompt change. Trajectory pays a substantial setup cost (47.2 s) but still spends a quarter of playback time stale, because reaching a leaf node forces on-the-fly extension; this leads to a worst-case latency of 23.7 s. The Trajectory approach therefore offers the worst of both worlds in this sequence: it requires precomputation comparable to parameter-space interpolation but does not eliminate user waiting.

Loop's worst-case latency of 8.7 s is roughly the duration of two buffers and reflects the case where a prompt change arrives just after a new buffer begins. This is not a failure of the strategy so much as an inherent feature of buffer-quantised generation. Loop's value is its zero setup cost, which makes it a reasonable default when responsiveness requirements are loose and the aesthetic of repetition is acceptable.

Comparing the Small and Medium results (Tables 1 and 2) reveals how each strategy's behaviour scales with model speed. Setup costs roughly double or triple, as expected. More interestingly, in-performance behaviour scales differently across strategies. Trajectory's Stale and Novel are identical between Small and Medium (24.7% and 63.0% in both): once the tree is built, navigation through it is independent of generation speed, and only the setup cost scales. Crossfade and Parameter Space behave similarly: their pre-generated streams or grid points fully decouple performance from generation speed, so Stale remains 0% and Novel remains 19.4% and 79.0% respectively. Granular's Novel drops from 57.9% to 47.4% on Medium, reflecting that its library regeneration cycle competes with playback and slower regeneration means more time spent on stale grains. Overlap collapses entirely on Medium: median latency of 25.1 s with novelty of just 6.7% indicates that the strategy's $R_g \geq S$ requirement is comprehensively violated, so the system spends nearly all of its playback time waiting for a buffer that arrives long after the prompt that requested it. This is the cleanest illustration in our evaluation of why scaffolding choice must be matched to model speed: a strategy that works well at one R_g can be unusable at another.

6 AESTHETIC AND PERFORMANCE CONSIDERATIONS

For looping, the resulting repetition-variation aesthetic echoes minimalist composition and live coding practices, suggesting that the constraint can be reframed as a stylistic feature. This approach works best with generation models that support explicit temporal structure conditioning to ensure loop coherence, or with more textural musical styles where loop boundaries are less perceptually salient.

Overlap provides middle ground between full buffer quantisation and continuous control. The technique works best for models showing quality degradation only at boundaries rather than throughout the buffer. Computational overhead of 2–6$\times$ excess generation may be acceptable in sufficiently resourced systems where responsiveness is paramount. As with crossfading, it works best for timbral changes where simple crossfades sound natural and phase relationships between sources are not perceptually critical.

With navigating trees, the paradigm shifts from continuous control to discrete musical decisions, suiting exploratory performance where compositional navigation constitutes the primary creative act. Exponential computational growth limits practical tree depth, requiring careful parameter trajectory design that balances expressiveness against computational feasibility.

Conversely, interpolation fundamentally shifts the model's role from source material generator to real-time synthesiser. This enables instrument-like control within pregenerated spaces but constrains exploration to the cache topology. The approach works best for parameter spaces with smooth interpolation properties and bounded ranges, similar to commercial preset morphing. Similarly, granular synthesis combines neural generation's timbral richness with granular synthesis's responsive control. Complex real-time manipulation becomes possible including grain density, duration, pitch transposition, and spatial position, while the generative model provides rich source material unavailable through traditional granular synthesis. This bridges neural generation and traditional performance techniques in a way that leverages the strengths of both approaches.

7 LIMITATIONS

Our empirical evaluation has several limitations that future work should address. First, we evaluate only a single model family (MusicGen, in two size variants) on a single GPU (RTX 2080) at one buffer size ($B = 4$ s). However, whilst more powerful computers and more efficient models exist, and will continue to be optimised, this problem this likely to continue for musicians without access to state-of-the-art hardware or models. Indeed, generation time decreases will likely always be traded against audio quality increases.

The Novel metric in particular is a property of the strategy combined with the performance, as discussed in the case study; absolute values are tied to our specific simulated interaction sequence, and a different rate or pattern of prompt changes would shift them.

All evaluation is computational. As we do not include performer studies, results are based on architectural reasoning rather than evidence from practitioners. We view the present work as establishing a vocabulary and a first empirical reference point, with cross-model, cross-hardware, and performer-grounded evaluation as natural next steps.

8 CONCLUSION

Real-time music generation faces two independent problems: buffer-quantised input yielding coarse temporal control and subreal-time generation yielding insufficient speed. We have presented a taxonomy of strategies organised by which constraint or constraints they address. Strategies for subreal-time generation including prebuffering, loop-based, and trajectory pregeneration enable performance with slow models but maintain buffer quantisation. Strategies for buffer quantisation including overlapping streaming and crossfading improve control granularity but require fast generation. Hybrid strategies including parameter space interpolation and granular-hybrid synthesis address both problems by decoupling real-time control from generation.

Our case study with MusicGen demonstrates that modest precomputation enables strategies that eliminate stale time entirely (crossfade and parameter-space interpolation), but the two strategies trade off content novelty against setup cost: crossfade buys responsiveness cheaply at the cost of recycled content, while parameter-space interpolation buys both responsiveness and novelty at substantially higher setup cost. The choice among strategies depends on available computational resources, acceptable setup time, and the specific aesthetic and interaction requirements of the performance context.

Whilst by no means exhaustive, by formalising the problems that many developers and performers regularly face, we hope to have aided the process of finding and comparing solutions. Indeed, this set of variables has been useful for the authors ourselves in categorising common solutions in the abstract, operationalising these as case studies and comparing the tradeoffs of the scaffolding approaches described within this paper.

Practitioners themselves can select appropriate strategies based on model constraints, computational resources, and performance requirements. Rather than treating buffer constraints purely as limitations to overcome, designers might consider them as structural properties that enable new forms of musical interaction adapted to discrete temporal frameworks. The repetition-variation aesthetics of loop-based approaches and the navigation-focused interaction of trajectory pregeneration suggest that these constraints can become compositional features when approached thoughtfully.

ETHICAL STATEMENT

Our case study uses MusicGen (Copet et al., 2023), which was trained on a corpus including licensed music. We acknowledge ongoing debates regarding the use of copyrighted musical material in training data for generative models, and the unresolved questions this raises for artists whose work may be represented in such corpora without explicit consent. Our contribution operates downstream of these training decisions: the scaffolding strategies presented are model-agnostic and could equally be applied to models trained on ethically sourced or artist-consented data. We encourage practitioners selecting models for performance to consider the provenance of training data as part of their decision criteria.

A central motivation of this work is that high-end hardware required to run state-of-the-art audio models at real-time speeds is not financially accessible to most musicians. The strategies presented are intended to broaden access by enabling performance on more modest equipment. We recognise that even the hardware used in our case study (an RTX 2080) remains beyond the means of many musicians globally, and we encourage further work targeting CPU-only or low-power deployment.

Tools that lower the barrier to incorporating generative audio in performance may shift creative labour in ways that affect working musicians. We frame these tools as scaffolding for human performers rather than replacements for them, and the strategies presented (loop-based composition, trajectory navigation, granular control) are explicitly designed to centre the performer's real-time agency. We do not advocate for the replacement of human musicians, session players, or sound designers.

This work did not involve human participants.

Code is released openly to support reproducibility and to allow practitioners to adapt the strategies to their own contexts and models, including those with more transparent training provenance.

The techniques presented enable more responsive deployment of generative audio models but do not introduce new generative capabilities. We do not believe the scaffolding strategies themselves create meaningful new misuse vectors beyond those already inherent to the underlying generative models.

REFERENCES

Anderson, D. P. and Kuivila, R. (1986). Accurately timed generation of discrete musical events. *Computer Music Journal*, 10(3):48–56.

Caillon, A. and Esling, P. (2022). Streamable neural audio synthesis with non-causal convolutions. In *Proc. 25th Int. Conf. Digital Audio Effects (DAFx-22)*, Vienna, Austria.

Caillon, A., McWilliams, B., Tarakajian, C., Simon, I., Manco, I., Engel, J., Constant, N., Li, Y., Denk, T. I., Lalama, A., Agostinelli, A., Huang, C.-Z. A., Manilow, E., Brower, G., Erdogan, H., Lei, H., Rolnick, I., Grishchenko, I., Orsini, M., Kastelic, M., Zuluaga, M., Verzetti, M., Dooley, M., Skopek, O., Ferrer, R., Petridis, S., Borsos, Z., van den Oord, Ä., Eck, D., Collins, E., Baldridge, J., Hume, T., Donahue, C., Han, K., and Roberts, A. (2025). Live Music Models. arXiv preprint https://arxiv.org/abs/2508.04651.

Caspe, F., Shier, J., Sandler, M., Saitis, C., and Mcpherson, A. (2025). Designing neural synthesizers for low latency interaction. *Journal of the Audio Engineering Society*.

Chew, E., Zimmermann, R., Sawchuk, A. A., Kyriakakis, C., Papadopoulos, C., François, A., Kim, G., Rizzo, A., and Volk, A. (2004). Musical interaction at a distance: Distributed immersive performance. In *Proceedings of the MusicNetwork Fourth Open Workshop on Integration of Music in Multimedia Applications*, pages 15–16. MusicNetwork Barcelona.

Copet, J., Kreuk, F., Gat, I., Remez, T., Kant, D., Synnaeve, G., Adi, Y., and Défossez, A. (2023). Simple and controllable music generation. *Advances in neural information processing systems*, 36:47704–47720.

Forsgren, S. and Martiros, H. (2022). Riffusion - stable diffusion for real-time music generation.

Hantrakul, L., Engel, J., Roberts, A., and Gu, C. (2019). Fast and flexible neural audio synthesis. In *Proc. 20th Int. Soc. Music Inf. Retrieval Conf. (ISMIR)*, pages 524–530, Delft, The Netherlands.

Jack, R. H., Stockman, T., and McPherson, A. (2016). Effect of latency on performer interaction and subjective quality assessment of a digital musical instrument. In *Proc. Audio Mostly 2016*, pages 116–123, New York, NY, USA. ACM.

Kamath, P., Gupta, C., and Nanayakkara, S. (2025). Morphfader: Enabling fine-grained controllable morphing with text-to-audio models. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE.

Lim, M., Knibbe, J., Chen, B., and Sima, Y. (2024). Echo Chamber: Generative Music AI within a Participatory Museum Sound Installation. *GenAICHI 2024: Generative AI and HCI at CHI 2024. Conference on Human Factors in Computing Systems - Proceedings*.

Roads, C. (1988). Introduction to granular synthesis. *Computer Music Journal*, 12(2):11–13.

Rottondi, C., Chafe, C., Allocchio, C., and Sarti, A. (2016). An overview on networked music performance technologies. *IEEE Access*, 4:8823–8843.

Scarlatos, A., Wu, Y., Simon, I., Roberts, A., Cooijmans, T., Jaques, N., Tarakajian, C., and Huang, C.-Z. A. (2025). ReaLJam: Real-Time Human-AI Music Jamming with Reinforcement Learning-Tuned Transformers. arXiv preprint https://arxiv.org/abs/2502.21267.

Schmid, A., Ambros, M., Bogon, J., and Wimmer, R. (2024). Measuring the just noticeable difference for audio latency. In *Proceedings of the 19th International Audio Mostly Conference: Explorations in Sonic Cultures*, pages 325–331.

Schwarz, D., Beller, G., Verbrugghe, B., and Britton, S. (2006). Real-time corpus-based concatenative synthesis with catart. In *9th International Conference on Digital Audio Effects (DAFx)*, pages 279–282.

Tralie, C. and Cantil, B. (2024). The concatenator: A bayesian approach to real time concatenative musaicing. arXiv preprint https://arxiv.org/abs/2411.04366.

Truax, B. (1988). Real-time granular synthesis with a digital signal processor. *Computer Music Journal*, 12(2):14–26.

Westerfeld, S. (2018). Spectmorph: Morphing the timbre of musical instruments. In *Linux Audio Conference 2018*, volume 10, page 5.