A Quantized Native Runtime for On-Device Semantic Audio Generation

Matteo Spanio
Centro di Sonologia Computazionale (CSC)
Dept. of Information Engineering, University of Padova
Padova, Italy
spanio@dei.unipd.it

Antonio Rodà
Centro di Sonologia Computazionale (CSC)
Dept. of Information Engineering, University of Padova
Padova, Italy
roda@dei.unipd.it

***Abstract*—Semantic audio applications increasingly require controllable generation on commodity and embedded hardware rather than through framework-heavy datacenter stacks. We present *aria*, a dependency-free native runtime that runs the complete text-to-music pipeline of Stable Audio 3 (SA3) on ordinary GPUs, CPU-only machines, and a Raspberry Pi 5, with no Python or deep-learning framework underneath. Our main contribution is a study of *quantization*: running the model at lower numerical precision to fit tight memory budgets, saving memory in place rather than adding to it. Because the runtime owns every internal tensor, it also exposes activation steering, a low-cost way to steer what the model generates. We judge the quality cost with three independent measures of the output (prompt adherence, overall audio quality, taste preservation), each compared against the ordinary variation between random seeds. Eight-bit precision shows no measurable quality loss on any measure while sharply cutting memory, and it is the fastest mode on the GPU; four-bit adds a small, bounded cost but shrinks the footprint enough to run the 1.2-billion-parameter model on an 8 GB Pi. Against the official implementation, aria matches or exceeds generation speed and starts about seven times faster. A case study of the steering interface generates music carrying taste associations (*sonic seasoning*), with genuine but bounded control for a subset of attributes. These results make a compact, quantized runtime with built-in control a practical basis for on-device semantic audio in Internet-of-Sounds settings. The *aria* runtime is released at https://github.com/matteospanio/aria.**

***Index Terms*—efficient inference, model quantization, edge computing, audio diffusion, music generation, activation steering**

I. Introduction

Semantic audio generation is moving from cloud demos toward interactive tools, local creative services, and embedded audio devices [1]. These settings care not only about generation quality but also about deployment properties: cold-start latency, predictable memory use, portability across commodity CPUs and GPUs, and the ability to keep a model resident near the user. Open-weight music models are now strong enough to be useful in such systems, yet their reference implementations still assume a Python/PyTorch serving stack and a discrete GPU. That mismatch matters for Internet-of-Sounds applications, where semantic audio often needs to run as a local or edge service rather than as a heavyweight datacenter job.

Local-inference work suggests a way forward, from the dependency-free llama.cpp family [2] to portable engines such as ONNX Runtime [3] and DwarfStar's port of datacenter-scale language models to user hardware [4]. The engineering lesson is general: once the serving stack is removed, model classes once treated as cloud-only can become practical on owned hardware.

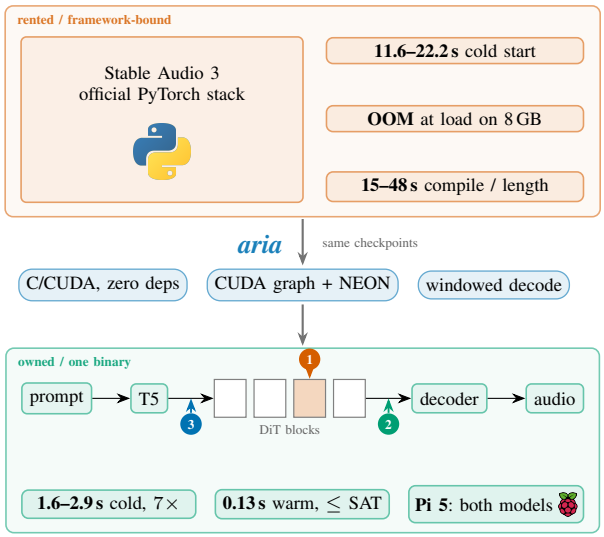

Fig. 1. With *aria*, the same Stable Audio 3 checkpoints move from a framework-bound serving stack (*top*) to a single dependency-free binary on hardware people own (*bottom*): 7× faster cold start, warm parity, and both model variants on a Raspberry Pi 5. Owning the pipeline also makes activation steering a runtime feature, injected at 1 the DiT residual stream, 2 the latent, or 3 the text conditioning.

Audio generation is a particularly strong candidate for this shift. Open music models are much smaller than frontier language models, roughly 1–10 B parameters, and diffusion inference scales with denoising steps rather than the far longer acoustic token sequences of autoregressive systems. The two leading open-source music generators, ACE-Step 1.5 [5] and Stable Audio [6], are both built on diffusion transformers (DiTs). We take Stable Audio 3 (SA3) as the reference target and ask a systems question: how much of its deployment cost belongs to the model itself, and how much belongs to the surrounding framework?

We answer that question with *aria*, a dependency-free C/CUDA runtime for SA3 that runs the full text-to-music pipeline, both model variants, on commodity GPUs and CPU-

only hardware (Figure 1). The runtime executes the tokenizer, text encoder, transformer denoiser, and autoencoder in a single native stack, stores weights from half precision down to 8- or 4-bit integers, runs the arithmetic in 8-bit where it pays off, and holds long-form generation within bounded memory. On a Raspberry Pi 5 (8 GB), the small model generates a 10 s stereo clip at a 1.9 GB full-precision peak, which quantization cuts to 0.84 GB at 8-bit. The 1.2 B-parameter medium model runs CPU-only at 4-bit ($\sim$200 s; 0.9 GB resident, 3.6 GB peak).

A native runtime also owns every intermediate tensor, which makes activation steering a runtime primitive rather than a Python-side patch or a retrained adapter: directions can be injected into the generation graph at negligible cost, giving semantic control while preserving the runtime's deployment advantages.

We use sonic seasoning, taste-conditioned music generation, as a case study for this capability [7], [8]. Taste is a useful stress test because it is difficult to specify lexically, grounded in an external crossmodal literature, and therefore informative about whether a learned steering target captures genuine semantic change or merely responds to degradation.

We make four contributions. (i) **The *aria* runtime**: a dependency-free C/CUDA engine that runs SA3's full pipeline (text encoder, transformer denoiser, autoencoder) on GPUs, CPU-only systems, and embedded-class hardware, with a calibrated efficiency study against the official implementation across warm latency, cold start, peak memory, and CPU-only generation. (ii) A **deployment quantization study**: weight storage from half precision down to 4-bit and an 8-bit-arithmetic mode on GPU tensor cores and ARM, made memory-frugal by releasing the full-precision weights once compressed. Rather than assume a fidelity budget, we measure the quality cost with three independent checks (prompt adherence, distributional quality, and taste preservation). On all three, 8-bit stays within re-seed noise, and 4-bit is sufficient, at a measurable but bounded cost, to put a 1.2 B model on an 8 GB Pi. (iii) An **in-graph activation-steering interface** with zero measurable overhead, bit-exact disable behaviour, and dose-response parity with the reference. (iv) A **bounded semantic-control case study** that uses sonic seasoning to test a hard-to-lexicalize attribute under a multi-oracle protocol. The result is a quantized native runtime with in-graph control, a practical basis for on-device semantic audio generation.

II. Related Work

A. Generative audio models

Text-to-music systems fall into two broad families: autoregressive transformers over discrete acoustic tokens, and latent-diffusion models that denoise in the latent space of a neural audio autoencoder. Diffusion cost scales with denoising steps rather than sequence length, and the leading open-weight systems, ACE-Step [5] and the Stable Audio family, are DiT-based. Our reference is Stable Audio 3 (SA3) [6]. It couples a SAME semantic-acoustic autoencoder [9], a 256-dimensional continuous latent, to a diffusion-transformer (DiT) denoiser [10] conditioned on text via T5Gemma [11] cross-attention and trained with flow matching [12]. We study its *small-music* (20 DiT blocks, d_{model}=1024) and *medium* (24 blocks, d_{model}=1536) configurations. The DiT residual stream of per-block hidden states is the substrate that activation steering reads and writes. The broader shift toward live generation, exemplified by Magenta / Lyria RealTime [13], makes warm, resident runtimes more attractive than cold per-request processes.

B. Generative models on edge devices

State-of-the-art generative audio increasingly runs behind closed APIs such as Suno[1] and Udio[2]. Open-weight reference stacks, by contrast, are typically heavyweight Python pipelines that assume a discrete GPU, and community CPU ports of SA3 remain tied to PyTorch [14]. The ggml lineage (llama.cpp [2], whisper.cpp [15], DwarfStar [4], stable-diffusion.cpp [16]) shows that a single, self-contained C/C++ program with memory-mapped weights, low-bit quantization, and hand-written vector and GPU kernels can deliver interactive inference on consumer hardware with near-zero cold start. Portability-first engines such as ONNX Runtime [3], ExecuTorch [17], Apple MLX [18], and NVIDIA TensorRT [19] trade model-specific kernels for generic graph execution. Neither approach has yet reached latent-diffusion music: to our knowledge, no dependency-free C runtime exists for a state-of-the-art music model, a gap that *aria* fills.

C. Quantization for on-device diffusion

Low-bit inference is the lever that makes these runtimes fit budget hardware. Post-training quantization (PTQ) compresses LLMs to 8-bit weights *and* activations without retraining [20], diffusion-specific PTQ pushes image models to 4-bit weights [21], and PTQ has recently been extended to audio diffusion transformers [22]. These works tune the quantizer for a fixed fidelity target. We instead address deployment. Our native runtime treats precision as a first-class axis, spanning half-precision, 8-bit, and 4-bit storage plus 8-bit arithmetic. By releasing the full-precision weights once compressed, it makes lower precision a memory *replacement* rather than an addition. It ships hardware-specific 8-bit kernels (integer tensor cores on the GPU, the matching instruction on ARM), and gates each precision against fp16 on independent objective checks of prompt adherence, distributional quality, and taste preservation.

D. Steering techniques

A frozen generative model can be controlled either by editing its weights or by intervening on its activations at inference time. The activation-based route rests on the linear representation hypothesis, according to which interpretable attributes are encoded approximately as linear directions in hidden space. Adding one vector to the residual stream can therefore shift an attribute while leaving the rest of the

[1]https://suno.com, accessed 2026-07-04.

[2]https://www.udio.com, accessed 2026-07-04.

computation intact. Activation addition [23] and representation engineering [24] extract such directions from contrastive examples and inject them at generation time. Tan et al. [25] characterize when these vectors generalize. This paradigm is now well represented in audio and music [26]–[31]. We take our injection design from Camporese et al. [26]. The weight-based family (LoRA [32], ControlNet [33], concept sliders [34]) can provide strong control, but it requires training, per-attribute data, and a growing parameter footprint. LoRA, the de-facto parameter-efficient standard, is our training-based comparator. Our case study targets a gustatory attribute, the five basic tastes: "sonic seasoning" reports reliable crossmodal associations between tastes and acoustic parameters [35], [36]. They are stable enough to shift the perceived taste of food [37], [38], and Spence later synthesized them under the label of gastrophysics [39]. These correspondences ground the reference used to assess steered generation; Section III gives the mapping.

III. Methodology

A. The aria runtime

aria carries the dependency-free native-runtime discipline of Section II over to audio latent diffusion. It targets the two settings in which the official Stable Audio 3 PyTorch stack is least practical: interactive or live generation from a resident service [13], and budget or CPU-only inference, where framework startup, GPU-context setup, and a multi-second cold start dominate short generations. *aria* is a single-purpose engine for SA3 in about 7.7k lines of C and CUDA, with no linear-algebra library, deep-learning framework, or other third-party dependency beyond the C math library and a threading runtime. It implements the whole pipeline behind a small dispatch layer: a text tokenizer, the T5Gemma text encoder [11], the diffusion transformer (DiT) that denoises the audio latent over eight sampling steps, and the audio autoencoder [9] (a 256-dimensional latent at $\sim$10.7 Hz). Supporting a new architecture adds one file and a registry entry, not a kernel rewrite. Weights are mapped directly from disk in half precision (fp16), with full precision (fp32) kept only where fidelity is sensitive. Optional lower-precision storage, down to 8- or 4-bit integers, keeps the medium model and small-memory GPUs within budget; once the compressed copy exists the runtime releases the original, so a lower precision *replaces* rather than adds to the footprint, cutting the Pi's peak memory from 1.9 to 0.84 GB at 8-bit.

One set of operations runs behind two backends. The CPU path is vectorized and multi-threaded, carries an 8-bit-integer matrix-multiply variant, and doubles as the correctness reference and the CPU-only production target. The GPU path uses half-precision tensor-core matrix multiplies and attention, including a banded decoder kernel; its per-step denoising loop is captured once and replayed as a single GPU graph. A windowed decoder caps memory at byte-identical output, and a streaming variant decodes only the frames it emits (3.1$\times$ faster per chunk on the Pi, still byte-identical); together they let the runtime generate and stream long-form audio on a Raspberry Pi 5. A resident batch mode and an HTTP server keep the model loaded across jobs for multi-client serving, roughly doubling throughput (0.60 vs. 1.23 s per job) at identical output.

a) Steering interface.: Because the runtime owns every intermediate tensor, activation steering becomes a built-in feature rather than a Python-side patch. *aria* loads precomputed direction vectors and applies them at three points in the pipeline, either adding the direction or projecting onto it and optionally only during a chosen span of sampling steps: the transformer's residual stream, the compact audio latent, and the text conditioning. This reproduces the reference intervention exactly ($\delta = \alpha\|h\|\,\hat{d}$; Section III). On the GPU the steering step lives inside the captured graph and reads a strength value refreshed each step (zero outside the chosen span), so enabling it triggers no re-capture and adds no measurable overhead; at strength 0 the output is bit-identical to the base model. The taste case study of Section III is thus a property of the runtime, not a fork of it.

b) Quantization and its evaluation.: The transformer's weights can be stored in half precision, or compressed to 8-bit or 4-bit integers (labelled q8 and q4). An optional mode also runs the arithmetic in 8-bit, quantizing activations alongside weights (W8A8), on the GPU's integer tensor cores or the equivalent ARM instruction. Because the compressed copy owns its memory, the runtime frees the full-precision source once packed, so a lower precision *reduces* resident memory rather than adding to it. We gate each precision empirically against the fp16 output on three independent objective checks (Section IV-B0a, Table II) rather than assuming a fidelity budget.

c) Efficiency protocol.: To compare deployments fairly, we separate three regimes. *Warm* generation, with the model already resident in-process on both sides, is the cost a live or batched service pays per request. *Invocation* is a one-shot command-line call that additionally re-pays per-process setup (text encoding and moving weights to the GPU). *Cold start* is process launch plus weight load plus one generation, the cost an edge endpoint pays on first use. We report all three on an RTX 3070 GPU and a CPU-only tier (Table I), analyzed in Section IV-B.

B. Taste steering and sonic seasoning

Sonic seasoning is the crossmodal finding that music systematically shifts perceived taste (consonant high-pitched legato reads as sweet, dissonant low music as bitter) across the five basic-taste axes $\{\mathsf{sweet}, \mathsf{sour}, \mathsf{bitter}, \mathsf{salty}, \mathsf{spicy}\}$ [35]–[38]. We ask whether a frozen music generator can be steered to *produce* audio carrying these associations. The machinery steers any difference-in-means direction. We choose taste as a difficult attribute that remains externally measurable: continuous, weakly lexicalized, and grounded in an independent psychophysics literature. We treat SA3 (Section III-A) as a black box and intervene on its activations. The taste oracle we optimize is susceptible to metric gaming, so a multi-oracle

protocol distinguishes genuine control from degradation that artificially raises the target.

1) Direction and injection site: Each DiT block writes its update additively, so its output lies in the same d_{model} space as its input. For axis i and block ℓ we estimate a unit direction $d_i^{(\ell)}$ pointing from "low-i" to "high-i" by difference-in-means, the simplest estimator known to transfer [23]–[26], over each contrastive set $S_i^{\pm}$:

$$d_i^{(\ell)} = \frac{\bar{h}_{i,\ell}^{+} - \bar{h}_{i,\ell}^{-}}{\left\|\bar{h}_{i,\ell}^{+} - \bar{h}_{i,\ell}^{-}\right\|_2}, \qquad \bar{h}_{i,\ell}^{\pm} = \frac{1}{|S_i^{\pm}|} \sum_{x \in S_i^{\pm}} \frac{1}{T} \sum_{t=1}^{T} h_\ell(x)_t. \tag{1}$$

The contrast comes from two sources. *Prompt-side* sets are caption pairs differing only in a taste descriptor, self-contained but encoding only what the text encoder associates with that word. *Audio-side* sets ground the contrast in the reference dataset [40], 377 music clips with per-clip basic-taste ratings. We rank these by rating on i and feed the top-/bottom-k clips to SA3 as an initial audio input, capturing the residuals it produces on audio humans actually heard as sweet or sour. Audio-side directions are more monotonic and more robust to over-steering, so we use them by default while retaining prompt-side directions as a controlled comparison.

2) Additive injection: At inference we add a scaled copy of the direction to the chosen block's output, for a single strength $\alpha \geq 0$,

$$h \;\leftarrow\; h + \alpha \, \|\bar{h}\| \, d_i^{(\ell)}, \tag{2}$$

where $\|\bar{h}\|$ is the mean baseline residual norm at ℓ, so α is in units of the typical residual magnitude, comparable across blocks and both model sizes. The intervention touches one block, adds one rank-one term, and has one hyperparameter. Setting $\alpha = 0$ leaves the pass unchanged and gives the matched, seed-identical baseline. We sweep α up to 1.0 because the response is non-monotonic: a useful regime exists only at low strength, while larger α severely degrades the audio.

3) Training-based baseline: per-axis LoRA: To quantify what training-free steering sacrifices, we train a per-axis LoRA [32] ($r = 8$) on the *same* contrasts with the backbone frozen, using a trigger-token objective (high-i clips captioned "i taste" vs. a neutral "music"). Unlike steering, it needs a per-axis optimization run, adds served parameters, and cannot be removed by zeroing a scalar. Both arms share the extraction data, pipeline, and evaluation, and each method's cost is logged (Section IV-C0g).

4) Multi-oracle evaluation: The central risk is circularity: the quantity we optimize is also the quantity one might be tempted to report as evidence of success. Our target is *wav2taste* [41], a learned audio-to-taste regressor. The effect on axis i is $\Delta_i(\alpha) = \hat{\tau}_i(x_\alpha) - \hat{\tau}_i(x_0)$ against the matched $\alpha = 0$ baseline. But wav2taste is imperfect (held-out macro Pearson $r \approx 0.67$; best sweet 0.82, worst sour 0.59) and can over-score out-of-distribution audio. We therefore guard it with three independent oracles: **CLAP** [42], [43] text–audio similarity to "X-tasting music" anchors, which shares no training signal, as a semantic cross-check; **Fréchet Audio Distance (FAD)** [44] over CLAP embeddings as a degradation detector; and **audio drift**, $1 - \cos(\phi(x_\alpha), \phi(x_0))$ on CLAP embeddings, as perturbation magnitude. All clips are loudness-normalized to -14 LUFS first, so no metric moves by loudness. We treat a setting as *genuine* only when Δ_i and CLAP rise together at low drift and bounded FAD. When Δ_i rises while CLAP falls and FAD or drift increase, we report degradation instead. A two-stage funnel locates this regime: a target-only per-layer scan *nominates* a candidate block per axis, then a dense α-window at that block is *selected* by all four oracles. The nomination is target-only by design. It is a cheap localizer, while the multi-oracle window is the actual gate, so the reported operating point is never chosen by the target alone. That ordering also exposes the failure mode. The scan ranks the final block first for the intense tastes, but the oracle panel then *overturns* that block as degradation (Section IV-C0b). A multi-oracle nomination would have hidden this rather than surfaced it. The candidate block is retained only if it survives the panel. Otherwise its clean operating point, or the axis, is rejected.

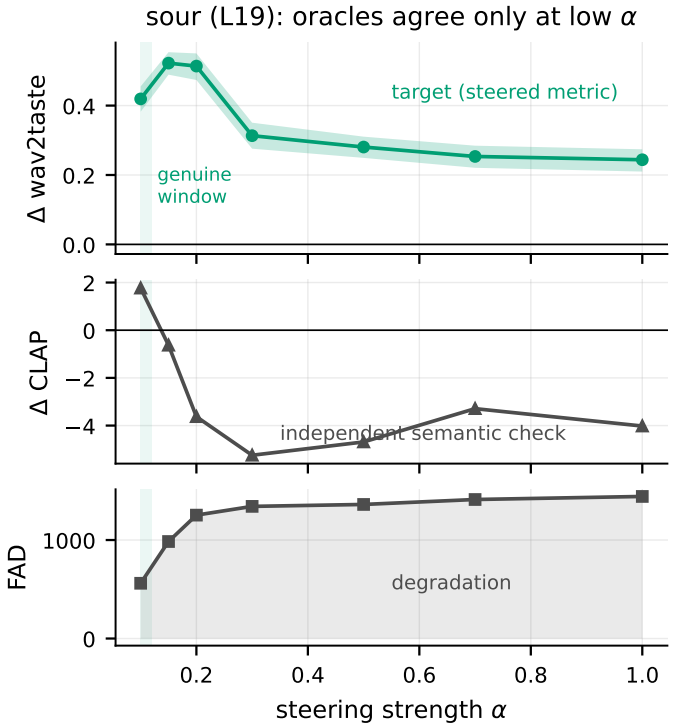

Fig. 2. The degradation demonstration (SOUR, small-music, its best layer L19), as two stacked panels sharing α (no dual axis). *Top:* the steering target (wav2taste Δ, with a 95% bootstrap CI band over the paired per-clip deltas) and the independent CLAP Δ on one axis; both rise inside the shaded low-α genuine window, then diverge as wav2taste stays high while CLAP collapses. *Bottom:* FAD explodes past the window. The high-α wav2taste gain is degradation that games the oracle. Colours are the fixed colourblind-safe taste palette (CLAP/FAD are shown neutral, being independent checks rather than tastes).

TABLE I

RUNTIME EFFICIENCY OF *aria* VERSUS THE OFFICIAL STABLE AUDIO 3 PYTORCH IMPLEMENTATION: 10 S CLIP, 8 SAMPLING STEPS, STEERING ACTIVE, ON AN RTX 3070 (8 GB) WITH AN I9-10900KF (CPU-ONLY ROWS USE 20 THREADS). *Warm* = GENERATION WITH THE MODEL ALREADY RESIDENT IN-PROCESS (BOTH SIDES); *invocation* = A ONE-SHOT COMMAND-LINE CALL THAT RE-PAYS PER-PROCESS SETUP; *cold* = PROCESS SPAWN + WEIGHT LOAD + ONE GENERATION. VRAM IS THE PEAK GPU PROCESS FOOTPRINT; "–" WHERE NOT APPLICABLE.

	small-music		medium	
Configuration	Time (s)	VRAM (MB)	Time (s)	VRAM (MB)
aria GPU (warm)	**0.13**	**1395**	**0.37**	**4215**
aria GPU (invocation)	1.23	–	2.55	–
aria (cold start)	**1.6**	–	**2.9**	–
aria CPU-only	2.5	–	9.8	–
SA3 GPU (warm)	0.146	2330	0.443	5948
SA3 (cold start)	11.58	–	22.23	–

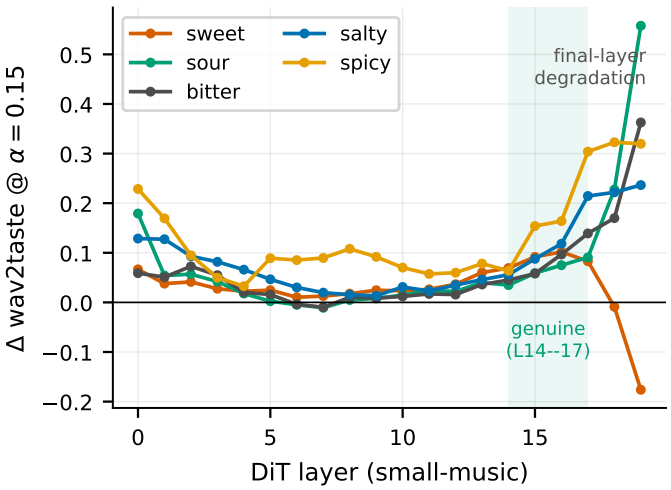

Fig. 3. Every DiT layer steered individually at α=0.15 (small-music; n=15 clips per point). SWEET peaks mid-late (L16) and collapses at the final blocks, while the four intense tastes spike at the final layer, a ranking the quality control later overturns as degradation (their genuine site is the shaded L14–17). Medium behaves alike, peaking at L20–21.

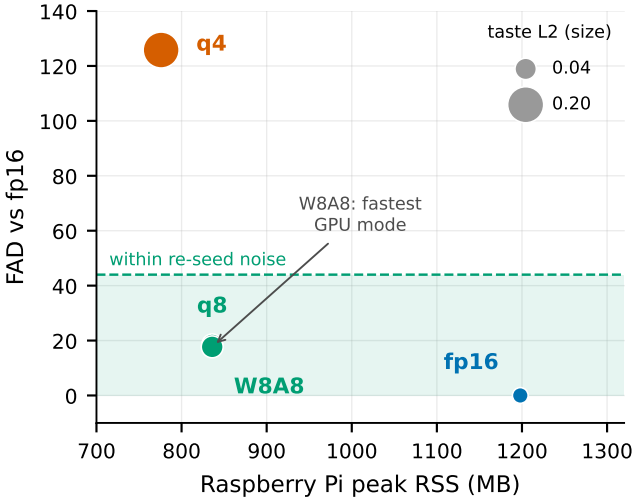

Fig. 4. Quantization performance in one space (small-music; all GPU device-resident, fp16 decoder, so only DiT precision differs). Quantizing slides a model left (less Raspberry-Pi memory) at a fidelity cost (up: FAD vs. the fp16 set); marker area grows with the wav2taste taste-vector L2. The shaded band is the FAD a mere fp16 re-seed induces. 8-bit (q8, W8A8) stays inside the band and small, and within the ΔCLAP and taste floors too (Table II), so it is indistinguishable from fp16; W8A8 is also the fastest GPU mode. 4-bit leaves the band for little extra memory but is what runs the 1.2 B medium model on an 8 GB Pi.

TABLE II

PRECISION LADDER (SMALL-MUSIC). TOP: DEPLOYMENT COST (GPU WARM 10 S AND PROCESS VRAM ON AN RTX 3070, RASPBERRY PI 5 PEAK MEMORY; MB). BOTTOM: FIDELITY VS. THE FP16 REFERENCE ON THE THREE OBJECTIVE CHECKS OF SECTION IV-A0E (ΔCLAP PROMPT ADHERENCE; FAD DISTRIBUTIONAL QUALITY IN A 32-DIMENSIONAL REDUCTION OF CLAP EMBEDDINGS; ΔTASTE, WAV2TASTE 5-D VECTOR). RE-SEED REFERENCE-NOISE FLOORS: ΔCLAP ±0.004, FAD 44.0, ΔTASTE 0.153. 8-BIT (Q8, W8A8) STAYS WITHIN EVERY FLOOR WHILE CUTTING VRAM ∼21% AND PI MEMORY TO 0.84 GB, AND W8A8 (8-BIT ARITHMETIC) IS THE FASTEST GPU MODE; 4-BIT CROSSES EVERY FLOOR BUT IS WHAT PUTS THE 1.2 B MEDIUM MODEL ON AN 8 GB PI. FP16 REPRODUCES FP32 (r=1.00).

	fp32	fp16	q8	W8A8	q4
GPU (s)	–	0.13	0.20	**0.10**	0.22
VRAM	–	1510	1190	1190	1128
Pi mem	1912	1198	836	836	776
ΔCLAP	–	ref	+0.002	−0.001	−0.020
FAD	–	ref	18.6	17.7	125.8
Δtaste	–	ref	0.035	0.044	0.199

IV. EXPERIMENTS AND RESULTS

A. Experimental setup

All conditions use 10 s clips, loudness-normalized to −14 LUFS before scoring, with three seeds $\{0, 1, 2\}$ and means reported. Steering strength α is expressed in units of the patched block's mean residual norm (Section III). Directions are grounded in *norm-sonic-seasoning* [40] (377 human-rated clips). We steer two released configurations: small-music (fp32, 20 blocks, d_{model}=1024) and medium (fp16, 24 blocks, d_{model}=1536). Medium in fp32 exceeds the 8 GB reference GPU. Precision therefore differs with size. To keep the two apart, we re-run small-music in *both* fp32 and fp16 (Section IV-B0a). The two dose–responses coincide (r=1.00), which isolates the medium–small differences as scale rather than precision. Per-point significance is assessed with a two-sided Wilcoxon signed-rank test on the n=36 paired deltas, Holm-corrected across the five axes.

TABLE III

DENSE-WINDOW MULTI-ORACLE STEERING (SMALL-MUSIC, FP32), EACH AXIS AT ITS WAV2TASTE-BEST LAYER. SWEET (L16) IS SHOWN ACROSS THE FULL α SWEEP; THE INTENSE TASTES (L19) AT α=0.1 VS. 0.15. IN EACH BLOCK THE TARGET (ΔW2T) AND THE INDEPENDENT CLAP CHECK RISE TOGETHER ONLY AT LOW α (THE GENUINE WINDOW); PAST IT CLAP FLIPS NEGATIVE (**BOLD**) AND FAD JUMPS WHILE WAV2TASTE KEEPS CLIMBING – METRIC-GAMING. SALTY BARELY MOVES CLAP EVEN WHEN CLEAN; SPICY IS MARGINAL.

Axis	α	Δw2t	ΔCLAP	FAD	drift
sweet (L16): full sweep					
sweet	0.1	+0.075	+1.05	73	0.11
	0.15	+0.097	+1.43	144	0.16
	0.2	+0.104	+1.83	274	0.25
	0.3	+0.119	+2.11	515	0.39
	0.5	+0.098	+0.82	777	0.52
	0.7	+0.056	**−1.16**	933	0.57
	1.0	−0.025	**−3.88**	1257	0.72
intense (L19): genuine at α=0.1					
sour	0.1	+0.419	+1.77	561	0.40
	0.15	+0.522	**−0.63**	985	0.62
bitter	0.1	+0.314	+1.66	836	0.54
	0.15	+0.360	**−1.15**	1222	0.73
salty	0.1	+0.209	+0.32	576	0.41
	0.15	+0.191	**−0.98**	885	0.57
spicy	0.1	+0.192	−0.18	301	0.26
	0.15	+0.267	+0.25	574	0.42

TABLE IV

THE *aria* RUNTIME REPRODUCES THE SA3/PYTORCH STEERING DOSE–RESPONSE: PER-AXIS PEARSON r AND MAE BETWEEN ARIA AND THE REFERENCE ON THE WAV2TASTE Δ (TARGET) AND CLAP Δ (INDEPENDENT CHECK), OVER THE SHARED PER-(AXIS, α) POINTS (n=35 PER MODEL). ARIA TRACKS THE TARGET SHAPE CLOSELY ON BOTH MODELS; THE LOOSER SMALL-MODEL CLAP AGREEMENT IS AN FP16-VS-FP32 NUMERICS OFFSET THAT DISAPPEARS ON MEDIUM. SPICY IS NOISIEST ON BOTH.

	wav2taste Δ		CLAP Δ	
Axis	r	MAE	r	MAE
small-music				
sweet	0.984	0.035	0.979	1.02
sour	0.962	0.040	0.985	1.42
bitter	0.979	0.022	0.973	1.57
salty	0.943	0.047	0.870	2.00
spicy	0.666	0.067	0.545	4.29
pooled	**0.953**	**0.042**	**0.859**	**2.06**
medium				
sweet	0.976	0.033	0.975	0.46
sour	0.949	0.040	0.921	1.06
bitter	0.916	0.037	0.971	1.40
salty	0.833	0.052	0.934	1.26
spicy	0.614	0.028	0.956	0.88
pooled	**0.916**	**0.038**	**0.946**	**1.01**

a) Extraction source and layer scan: We compare prompt- vs. audio-side directions by dose-response monotonicity (Spearman ρ) and over-steer robustness, holding block, grid, and seeds fixed. To localize *where* taste is steerable, a logistic probe identifies the layers at which the axes are linearly separable. We then steer every DiT block at $\alpha = 0.15$ (5 prompts $\times$ 3 seeds). Blocks are ranked by the target metric alone. One candidate block per axis is then submitted to the oracle panel for confirmation or rejection.

b) Dense window and oracle panel: At each axis's best layer on small-music we sweep $\alpha \in \{0.1, 0.15, 0.2, 0.3, 0.5, 0.7, 1.0\}$ over 12 prompts $\times$ 3 seeds, scoring every clip with the four-oracle panel of Section III (wav2taste target, CLAP cross-check, FAD, drift). A model-size study repeats the sweet sweep on medium, with each model evaluated at its own re-scanned best layer.

c) Ablation designs: Three ablation families reuse the dense-window panel on small-music. *Injection op*: besides additive steering we test projection, amplifying (β>0) or removing (β<0) the direction component already present per token, swept both ways. *Step window*: injection confined to the early (0–3) vs. late (4–7) denoise steps at matched strength. *Steering site*: the same difference-in-means recipe applied to the compact audio latent and the pooled text embedding, comparing residual, latent, and text injection. The training comparator is the per-axis rank-8 LoRA of Section III (800 steps), dose-swept and scored with the same panel.

d) aria reproduction and efficiency: To confirm the runtime preserves steering behaviour, we re-run the dense window through *aria* with the identical exported direction vectors and α grid, and report per-axis Pearson r and MAE against the SA3/PyTorch reference on the wav2taste and CLAP dose-responses. Efficiency is measured on the RTX 3070 and a CPU-only x86 tier under the warm / invocation / cold protocol of Section III-A (Table I). The baseline is the official Stable Audio 3 implementation (commit `dedace19`, 2026-07-01; torch 2.7.1, CUDA 12.6), run in half precision with its native 8-step sampler and no classifier-free guidance. With the faster attention kernel unavailable, the repository's fallback attention path is active, and reported memory is the process's GPU footprint. The repository disables several standard performance options by default; we measured both that default and a tuned configuration (those options enabled, plus compiling the transformer) and cite the *faster* one throughout.

e) Quantization fidelity: We characterize each precision against fp16 rather than fixing a fidelity budget. Every clip is generated on the GPU with the decoder held at half precision, so a comparison isolates the transformer's weight (q8, q4) or weight-plus-activation (W8A8) quantization from any device effect. We use 24 genre-diverse music prompts $\times$ 3 seeds (72 clips per precision, 10 s, loudness-normalized). We score three metrics drawn from two independent audio models. Two come from CLAP [43], a text–audio embedding model: *prompt adherence* (how closely each clip matches its own text prompt) and *distributional quality* (Fréchet Audio Distance, FAD [44], between the precision's clips and the fp16 set in

TABLE V

CONSOLIDATED OPERATING POINTS FOR THE INJECTION-OP, STEP-WINDOW, STEERING-SITE, AND METHOD ABLATIONS (SMALL-MUSIC, FP32; n=36 CLIPS PER POINT, 12 PROMPTS × 3 SEEDS). ΔW2T IS THE WAV2TASTE TARGET SHIFT VS. THE MATCHED α=0 BASELINE; ΔCLAP IS THE INDEPENDENT SEMANTIC CHECK (**BOLD** ONCE IT HAS FLIPPED NEGATIVE, I.E. OFF THE GENUINE WINDOW); FAD IS DEGRADATION VS. BASELINE. THE FOUR GROUPS ARE READ IN THE STEERING RESULTS (SECTION IV-C).

Axis	Method / site	Op-point	Δw2t	ΔCLAP	FAD
Injection op (DiT residual)					
sour	additive	α=0.1	+0.419	+1.77	561
	additive	α=0.15	+0.522	**−0.63**	985
	projection	β=1	+0.492	+0.60	544
sweet	additive	α=0.3	+0.119	+2.11	515
Denoise-step window (DiT residual, sour)					
sour	early (steps 0–3)	α=0.2	+0.530	**−4.26**	1294
	late (steps 4–7)	α=0.2	+0.515	+0.80	565
Steering site					
sour	latent (256-D)	scale 1	+0.385	+1.34	365
sweet	latent (256-D)	scale 0.5	+0.050	+0.69	197
sour	text (768-D)	scale 0.5	+0.299	**−3.04**	935
Method: LoRA vs. best steering					
sour	LoRA	α=1.0	+0.341	**−0.92**	767
sweet	LoRA	α=0.5	+0.032	**−0.10**	171
sour	steering	proj. β=1	+0.492	+0.60	544
sweet	steering	add. α=0.3	+0.119	+2.11	515

a 32-dimensional embedding space, reduced for stability at this sample size). The third, from the separate wav2taste [41] model, is *taste preservation* (the distance between the 5-D taste vectors of each clip and its fp16 counterpart). Re-seeding fp16 (seeds 3–5) gives each metric a reference-noise floor, the change a re-seed alone induces; a precision is within noise below it.

B. Runtime benchmarks

Table I reports the three deployment regimes across both tiers. In the warm setting, with the model resident on both sides, *aria* slightly edges the official implementation after a round of kernel tuning and request reuse. A steered 10 s clip runs in 0.13 vs. 0.146 s (small-music) and 0.37 vs. 0.443 s (medium). It cold-starts 7.2–7.7× faster and holds a 1.4–1.7× smaller peak GPU-memory footprint (Table I), and its resident batch mode amortizes per-process setup to 0.60 s per job. The runtime also operates without a GPU. After a pass that parallelizes and vectorizes the elementwise work between the matrix multiplies, a 10 s clip decodes in 2.5 s on small-music (∼4× real time) and 9.8 s on medium on a 20-thread CPU, and chunked sliding-window generation streams at a real-time factor of 0.6–4× on the RTX 3070. A calibrated fast preset (six denoising steps instead of eight) trims a further ∼25% off every transformer-bound path, with a taste-shift below the seed-to-seed noise floor on both tiers.

At 60 s on GPU the picture inverts on the medium tier. A half-precision attention output stage and running the mapping convolution as a matrix multiply put *aria* at 1.28 s against the reference's fused-attention path at 1.38 s, faster at equal precision. An opt-in 8-bit-arithmetic mode (W8A8, of 4-bit-class fidelity) extends the lead to 1.12 s. Only on small-music, where self-attention dominates the smaller autoencoder, does the official path stay ahead (0.52 vs. 0.38 s, 1.37×). CPU favours *aria* too: its banded decoder renders 60 s of medium 1.68× faster (48.0 vs. 80.65 s), and the small-tier CPU gap to the PyTorch baseline essentially closes (1.09×). The official warm figure relies on the reference's kernel compiler, which recompiles for *each* new clip length (a 14.5–48.1 s penalty, poorly suited to varying lengths); its medium loader also briefly needs ∼7.1 GB, overflowing an 8 GB card, so the benchmark used a low-memory load path.

a) Quantization is close to free at 8-bit, and enables the edge: Precision is a deployment axis, so we characterize rather than fix it (Table II, Figure 4) with the three checks of Section IV-A0e. To avoid validating quantization with the same oracle the steering study optimizes, we lead with the two msclap-based metrics that share no training path with the taste oracle: prompt adherence and FAD. We keep the network-independent wav2taste L2 as a narrow corroborator, and scale each metric by the change a fp16 re-seed induces. Eight-bit shows no measurable degradation on any of the three. Both q8 and W8A8 stay inside every re-seed floor while cutting GPU memory ∼21% and Pi resident memory 2.3×, and W8A8 is the fastest GPU mode (0.10 s warm, on the GPU's integer tensor cores). Four-bit measurably crosses every floor, most on prompt adherence (5× its floor). It nonetheless lets the 1.2 B medium model run on the 8 GB Pi (∼200 s); fidelity here is small-music, and the medium case is a memory-fit result. fp16 reproduces the fp32 dose–response (r=1.00, MAE 0.003; steering sweep), so precision is not confounded with the model-scale results below.

b) Live steering: Because steering runs inside the graph and the model stays warm, the control can be turned on during a stream. With the SWEET scale ramped 0→0.5→0 across a 12-chunk stream, the per-chunk taste score tracks the schedule (Spearman ρ=0.78, p=0.003), lagging on the descent as the continuation inherits context. On the Raspberry Pi 5, a steered clip costs the same as an unsteered one (32.9 vs. 32.8 s), so

semantic control remains fully on-device.

C. Steering results

Extraction source. On small-music, the prompt-derived SWEET direction yields an inverted-U response, peaking at $\Delta_{\text{w2t}}{=}{+}0.05$ ($\alpha{=}0.1$) then collapsing to -0.08 at $\alpha{=}0.2$; the audio-derived direction reaches the same peak but remains monotonic (Spearman $\rho{=}{+}0.94$) and still positive at $\alpha{=}0.2$. We therefore use audio-side directions throughout.

a) A narrow clean window: The central result is that wav2taste alone is not a reliable guide. Sweeping α at the SOUR axis's best layer L19 (Figure 2, Table III), the oracles agree only at low strength. At $\alpha{=}0.1$ the shift is large ($\Delta_{\text{w2t}}{=}{+}0.42$, $p{=}2.9{\times}10^{-11}$, $d{=}3.71$) with CLAP positive, genuine steering. Past $\alpha{=}0.1$ they diverge: wav2taste keeps climbing to its maximum at $\alpha{=}0.15$ while CLAP flips negative and FAD jumps. By $\alpha{=}1.0$ the audio has collapsed into noise-like output that wav2taste, weakest on sour ($r{\approx}0.59$), misreads as intense taste. The metric's maximum lies *inside* the degradation regime. In the clean window, SWEET, SOUR, and BITTER steer genuinely with CLAP confirmation, whereas SALTY and SPICY stay weak even when clean; we therefore claim usable control only for the first three axes and report the case study as a bounded proof of concept, not broad taste controllability. On ten held-out prompts never used for layer or strength selection, both operating points replicate (SWEET ${+}0.164$, SOUR ${+}0.508$; Holm $p{=}3.7{\times}10^{-9}$). An *audio*-anchored CLAP variant (similarity to the centroid of the top-taste clips instead of a text anchor) reproduces the same boundary (SOUR ${+}0.12$ at $\alpha{=}0.1$, negative from $\alpha{=}0.3$). The window is therefore neither a layer-selection artefact nor a text-anchoring artefact.

b) Layer selection depends on the attribute: Linear separability does not imply steerability. A logistic probe places taste in early text-encoding layers. Yet steering each block at $\alpha{=}0.15$ (Figure 3) shows SWEET peaks mid-late at L16 and collapses at the final blocks, while the intense tastes rank the *final* block first. Independent quality checks overturn that ranking: the scan-winning values already sit in the degradation regime with CLAP negative, so layer selection cannot be delegated to the target. The most reliable operating point is SWEET at L16, a modest $\Delta_{\text{w2t}}{=}{+}0.119$ at $\alpha{=}0.3$ ($p{=}3.2{\times}10^{-9}$, $d{=}1.20$) with CLAP ${+}2.11$ at bounded FAD (Table III).

c) Model scale: A larger backbone widens the clean operating window rather than merely inflating the target metric. For SWEET, medium reaches a higher clean peak ($\Delta_{\text{w2t}}{=}{+}0.143$ at $\alpha{=}0.15$; $p{=}8.9{\times}10^{-9}$, $d{=}1.32$) and holds a positive CLAP plateau farther out than small-music. At $\alpha{=}0.1$ medium also degrades far less on the intense tastes (lower FAD across axes) while keeping CLAP positive. Re-running the small model's full window in fp16 reproduces its fp32 dose–response (pooled $r{=}1.00$, MAE 0.003 over all five axes), so the medium–small differences are scale, not precision.

d) aria reproduces the dose–response: Re-running the dense window through *aria* reproduces the reference closely (Table IV): pooled wav2taste $r{=}0.95$ (small) and 0.92 (medium), with SWEET tightest. CLAP agrees on medium ($r{=}0.95$) but is looser on small ($r{=}0.86$), reflecting a numerical offset (aria runs in half precision where the small reference runs in full) that vanishes on medium. Both the genuine and degradation regimes carry over, so the case-study conclusions hold.

e) Ablations: injection operator and step window: Two ablations run entirely through the runtime's batch mode (the PyTorch injector supports neither), consolidated in Table V. *Op*: projection *amplification* (scaling up the direction already present in each token) matches additive steering's peak SOUR shift at markedly lower FAD and with CLAP still positive ($\beta{=}1$: $p{=}2.9{\times}10^{-11}$, $d{=}4.08$), but it does nothing for SWEET, whose residual component is too small to amplify. The two operators are complementary across axes. Steering *away* from the direction has little effect on either axis. *Window*: confining injection to the *late* denoise steps (4–7) roughly halves the damage at matched taste shift (late $\alpha{=}0.2$: $p{=}2.9{\times}10^{-11}$, $d{=}5.10$, CLAP positive, vs. the early steps' CLAP collapse), consistent with early steps setting global structure while late steps shape the timbre both oracles read.

f) Steering sites vary by attribute: The same protocol evaluates the pipeline's three injection sites (Table V). The compact-*latent* site steers SOUR cleanly, a better quality-per-shift point than the residual site (scale 1: $p{=}2.9{\times}10^{-11}$, $d{=}3.11$, CLAP positive at low FAD), but fails SWEET, which inverts. The *text-embedding* site fails both axes at *every* scale tested (0.5–32): even the smallest nudge reads as degradation and SWEET turns negative. This suggests that a mean shift in the text-embedding space moves the conditioning off the data manifold rather than encoding "more taste", and corroborates the prompt-side extraction failure at the embedding level. SOUR, an overtly timbral attribute, is steerable at latent and residual sites alike, while SWEET responds only to mid-late residual injection.

g) Steering versus LoRA: Under the same evaluation panel, the trigger-token LoRA (rank 8, 800 steps) never reaches a CLAP-positive operating point on either axis (Table V): SOUR peaks with CLAP negative, below steering's clean shift, and SWEET barely moves ($p{=}0.16$, n.s.), turning *negative* at higher strength. A larger, longer-trained adapter (rank 32, $3{\times}$ the steps) lands in the same place. The cost comparison is one-sided too: LoRA needs ${\sim}5$ min of training, 5.15 M parameters, and 10.4 MB per axis, whereas steering trains nothing, stores a 4–6 KB vector, runs in-graph at no overhead, and is removable at serve time. For this lexically ill-defined attribute, parameter updates add no control over a simple steering vector: steering wins on effect and cleanliness, not only on cost.

V. Discussion

A. Implications of warm-parity edge deployment

On warm throughput the two stacks are within a few percent, with *aria* marginally ahead: the runtime gains less from faster arithmetic than from removing framework startup,

GPU-context setup, and per-length compilation, the costs that dominate short generations. Interactive and embedded settings benefit most from a resident, warm model, since cold start, memory footprint, and predictable behaviour across clip lengths matter more there than steady-state speed. The one remaining GPU deficit is long-form generation on the *small* tier; on medium *aria* overtakes the official path at exact precision, and CPU-only it renders the same clip faster. Quantization is what makes the embedded tier practical: 8-bit shows no measurable degradation on any of the three checks, releasing the full-precision copy turns lower precision into a memory *reduction* that reaches an 8 GB Pi at 4-bit, and on the GPU 8-bit arithmetic is the fastest mode, so compression and speed do not trade off.

B. Learned oracles require independent checks

When an intervention is tuned to maximize a learned metric, that metric ceases to be a faithful indicator once the intervention becomes too aggressive: the target keeps rising while causally independent checks collapse. The same limitation applies to layer selection, because the target can mistake degradation for success. Both the strength and the site of an intervention must therefore be supervised by quality signals that share no training path with the target. This caution extends to any representation-editing evaluation that reports the optimized quantity as its own evidence.

C. Scope and limitations

The systems contributions (the runtime, the quantization study, the efficiency comparison) rest on direct measurement and stand on their own. The steering study is scoped as a bounded case study. It intervenes on a single model family with learned or distributional oracles, so we claim genuine control only for the three attributes where independent checks confirm it and treat the rest as a negative result. Those operating points replicate on held-out prompts, survive an audio-anchored re-test, and are stable across a precision change (half versus full), the runtime port, and a stronger adapter (Section IV-C). What remains open is perceptual confirmation, since no automatic oracle equals human listening. A listening study is therefore the natural next step rather than a gap in the present claims, which are pitched at exactly what the oracles can and cannot certify.

VI. Conclusion

We presented *aria*, a dependency-free C/CUDA runtime for Stable Audio 3, to show that a state-of-the-art latent-diffusion music model can be deployed without a datacenter-oriented serving stack. In roughly 7.7k lines of C, with no framework or third-party dependencies, aria runs the full text-to-music pipeline on a commodity CPU and an inexpensive GPU: it matches the official implementation's warm GPU throughput (slightly exceeding it after kernel tuning), cuts cold start by about 7×, lowers peak GPU memory, and runs CPU-only at ∼4× real time, with long-form GPU generation on the small tier its one remaining limitation. Treating precision as a deployment axis, 8-bit weights leave prompt adherence, distributional quality, and taste all within re-seed noise, 8-bit arithmetic runs fastest on the GPU, and releasing the full-precision weights once compressed puts the 1.2 B model on an 8 GB Pi at 4-bit. Its bit-exact activation-steering interface reproduces the reference dose–response, and the sonic-seasoning case study shows that taste control is real but confined to a narrow operating window for a subset of attributes. The wider lesson is that semantic control should be evaluated with independent quality checks, not with the optimized metric alone.

A. Outlook

Several extensions are clear. Fusing the decoder's kernels has reversed the medium-tier long-form gap, so *aria* overtakes the official path there at exact precision. The remaining small-tier distance comes from the transformer's full self-attention, whose attention weights we measure to be too spread out for a banded approximation; a fused attention kernel (FlashAttention-class) behind a build flag is the open lever. A persistent server mode would keep the model resident for multi-client serving and support latent-domain streaming continuation [13]. The main open validation is perceptual. As future work we plan a human listening study, using pairwise comparisons fit with a Bradley–Terry model [45], to externally confirm the automatic oracles on the attributes where they indicate genuine control. Both SA3 variants run on a Raspberry Pi 5, the medium only at 4-bit, where the 8-bit-arithmetic path is the fastest CPU mode.

Acknowledgment

This work was funded by the European Union - NextGenerationEU, under the National Recovery and Resilience Plan (PNRR).

References

[1] Z. Evans, J. D. Parker, C. Carr, Z. Zukowski, J. Taylor, and J. Pons, "Stable audio open," in *ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2025, pp. 1–5.

[2] D. Gope, D. Mansell, D. Loh, and I. Bratt, "Highly optimized kernels and fine-grained codebooks for llm inference on arm cpus," 2024. [Online]. Available: https://arxiv.org/abs/2501.00032

[3] O. R. developers, "Onnx runtime," https://onnxruntime.ai/, 2021, version: 1.27.0.

[4] S. Sanfilippo and contributors, "Dwarfstar," https://github.com/antirez/ds4, 2026, deepSeek 4 Flash and PRO local inference engine for Metal, CUDA and ROCm.

[5] J. Gong, Y. Song, W. Zhao, S. Wang, S. Xu, J. Guo, and X. Yang, "Ace-step 1.5: Pushing the boundaries of open-source music generation," 2026. [Online]. Available: https://arxiv.org/abs/2602.00744

[6] Z. Evans, J. D. Parker, M. Rice, C. Carr, Z. Zukowski, J. Taylor, and J. Pons, "Stable audio 3," 2026. [Online]. Available: https://arxiv.org/abs/2605.17991

[7] M. Spanio, M. Zampini, A. Rodà, and F. Pierucci, "A multimodal symphony: integrating taste and sound through generative ai," *Frontiers in Computer Science*, vol. Volume 7 - 2025, 2025.

[8] M. Spanio, "Towards emotionally aware ai: Challenges and opportunities in the evolution of multimodal generative models," in *Proceedings of the AIxIA Doctoral Consortium 2024 co-located with the 23nd International Conference of the Italian Association for Artificial Intelligence (AIxIA 2024)*, ser. CEUR Workshop Proceedings. http://CEUR-WS.org, 2024. [Online]. Available: https://ceur-ws.org/Vol-3914/

[9] J. D. Parker, Z. Evans, C. Carr, Z. Zukowski, J. Taylor, M. Rice, and J. Pons, "Same: A semantically-aligned music autoencoder," 2026. [Online]. Available: https://arxiv.org/abs/2605.18613

[10] W. Peebles and S. Xie, "Scalable diffusion models with transformers," in *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct 2023, pp. 4172–4182.

[11] G. Team, M. Riviere, S. Pathak, P. G. Sessa, C. Hardin, S. Bhupatiraju, L. Hussenot, T. Mesnard, B. Shahriari, A. Ramé *et al.*, "Gemma 2: Improving open language models at a practical size," 2024. [Online]. Available: https://arxiv.org/abs/2408.00118

[12] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow Matching for Generative Modeling," in *International Conference on Learning Representations*, 2023. [Online]. Available: https://mlanthology.org/iclr/2023/lipman2023iclr-flow/

[13] A. Caillon, B. McWilliams, C. Tarakajian, I. Simon, I. Manco, J. Engel, N. Constant, Y. Li, T. I. Denk, A. Lalama, A. Agostinelli, C.-Z. A. Huang, E. Manilow, G. Brower, H. Erdogan, H. Lei, I. Rolnick, I. Grishchenko, M. Orsini, M. Kastelic, M. Zuluaga, M. Verzetti, M. Dooley, O. Skopek, R. Ferrer, Z. Borsos, A. van den Oord, D. Eck, E. Collins, J. M. Baldridge, T. Hume, C. Donahue, K. Han, and A. Roberts, "Live music models," in *The Thirty-ninth Annual Conference on Neural Information Processing Systems Creative AI Track: Humanity*, 2025. [Online]. Available: https://openreview.net/forum?id=SB3XbPgvpl

[14] innermost47, "Obsidian-neural: Stable audio 3 medium running locally on cpu," https://github.com/innermost47/ai-dj, 2026, local edition; reports ≈11 s/generation on a laptop CPU (Apple Silicon). Verify before camera-ready.

[15] G. Gerganov and contributors, "whisper.cpp: Port of openai's whisper model in c/c++," https://github.com/ggml-org/whisper.cpp, 2022, ggml-based, dependency-free speech recognition on CPU and GPU.

[16] J. Ng, C. Lv, P. Zhao, W. Niu, J. Lin, M. Pan, Y. Liang, and Y. Wang, "Open-source acceleration of stable-diffusion.cpp deployable on all devices," 2025. [Online]. Available: https://arxiv.org/abs/2412.05781

[17] M. Nachin, D. Desai, S. S. Jia, C. Lai, M. Liu, J. Szwejbka, R. Alvarez, R. Ascani, D. Bort, M. Candales *et al.*, "ExecuTorch - a unified PyTorch solution to run AI models on-device," *arXiv preprint arXiv:2605.08195*, 2026. [Online]. Available: https://github.com/pytorch/executorch

[18] A. Hannun, J. Digani, A. Katharopoulos, and R. Collobert, "MLX: Efficient and flexible machine learning on apple silicon," 2023. [Online]. Available: https://github.com/ml-explore

[19] NVIDIA Corporation. (2018) Tensorrt. https://developer.nvidia.com/tensorrt. Per-model engine compilation for datacenter GPU inference. [Online]. Available: https://github.com/NVIDIA/TensorRT

[20] G. Xiao, J. Lin, M. Seznec, H. Wu, J. Demouth, and S. Han, "SmoothQuant: Accurate and efficient post-training quantization for large language models," in *International Conference on Machine Learning (ICML)*, 2023.

[21] X. Li, Y. Liu, L. Lian, H. Yang, Z. Dong, D. Kang, S. Zhang, and K. Keutzer, "Q-Diffusion: Quantizing diffusion models," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[22] T. Khandelwal and M. Fuentes, "Post-training quantization for audio diffusion transformers," 2025.

[23] A. M. Turner, L. Thiergart, D. Udell, G. Leech, U. Mini, and M. MacDiarmid, "Activation addition: Steering language models without optimization," *CoRR*, vol. abs/2308.10248, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2308.10248

[24] A. Zou, L. Phan, S. Chen, J. Campbell, P. Guo, R. Ren, A. Pan, X. Yin, M. Mazeika, A.-K. Dombrowski, S. Goel, N. Li, M. J. Byun, Z. Wang, A. Mallen, S. Basart, S. Koyejo, D. Song, M. Fredrikson, J. Z. Kolter, and D. Hendrycks, "Representation engineering: A top-down approach to ai transparency," 2025. [Online]. Available: https://arxiv.org/abs/2310.01405

[25] D. Tan, D. Chanin, A. Lynch, B. Paige, D. Kanoulas, A. Garriga-Alonso, and R. Kirk, "Analysing the generalisation and reliability of steering vectors," in *Proceedings of the 38th International Conference on Neural Information Processing Systems*, ser. NIPS '24. Red Hook, NY, USA: Curran Associates Inc., 2024.

[26] G. Camporese, "Mood vectors in audio diffusion: Steering stable audio 3," May 2026. [Online]. Available: https://guglielmocamporese.github.io/blog/audio-mood-steering.html

[27] S. Facchiano, G. Strano, D. Crisostomi, I. Tallini, T. Mencattini, F. Galasso, and E. Rodolà, "Activation patching for interpretable steering in music generation," 2025. [Online]. Available: https://arxiv.org/abs/2504.04479

[28] Ł. Staniszewski, K. Zaleska, M. Modrzejewski, and K. Deja, "TADA! tuning audio diffusion models through activation steering," in *ICLR 2026 Workshop on Representational Alignment (Re^4-Align)*, 2026. [Online]. Available: https://openreview.net/forum?id=OUux4upENk

[29] D. Zhao, D. Beaglehole, J. McAuley, T. Berg-Kirkpatrick, and Z. Novack, "Steering autoregressive music generation with recursive feature machines," in *The Fourteenth International Conference on Learning Representations*, 2026. [Online]. Available: https://openreview.net/forum?id=NaHzPMaCY9

[30] N. Singh, M. Cherep, and P. Maes, "Discovering and steering interpretable concepts in large generative music models," in *AI for Music Workshop*, 2025. [Online]. Available: https://openreview.net/forum?id=jVSlJk5qNA

[31] N. Paek, Y. Zang, Q. Yang, and R. Leistikow, "Learning interpretable features in audio latent spaces via sparse autoencoders," in *Mechanistic Interpretability Workshop at NeurIPS 2025*, 2025. [Online]. Available: https://openreview.net/forum?id=5fsYFQzzMX

[32] E. J. Hu, yelong shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-rank adaptation of large language models," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=nZeVKeeFYf9

[33] L. Zhang, A. Rao, and M. Agrawala, "Adding conditional control to text-to-image diffusion models," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2023, pp. 3836–3847.

[34] R. Gandikota, J. Materzyńska, T. Zhou, A. Torralba, and D. Bau, "Concept sliders: Lora adaptors for precise control in diffusion models," in *Computer Vision – ECCV 2024*, A. Leonardis, E. Ricci, S. Roth, O. Russakovsky, T. Sattler, and G. Varol, Eds. Cham: Springer Nature Switzerland, 2025, pp. 172–188.

[35] A.-S. Crisinel and C. Spence, "As bitter as a trombone: Synesthetic correspondences in nonsynesthetes between tastes/flavors and musical notes," *Attention, Perception, & Psychophysics*, vol. 72, no. 7, pp. 1994–2002, 2010. [Online]. Available: https://doi.org/10.3758/APP.72.7.1994

[36] B. Mesz, M. A. Trevisan, and M. Sigman, "The taste of music," *Perception*, vol. 40, no. 2, pp. 209–219, 2011.

[37] K. Knöferle and C. Spence, "Crossmodal correspondences between sounds and tastes," *Psychonomic Bulletin & Review*, vol. 19, pp. 992–1006, 2012.

[38] Q. J. Wang, A. T. Woods, and C. Spence, ""what's your taste in music?" a comparison of the effectiveness of various soundscapes in evoking specific tastes," *i-Perception*, vol. 6, no. 6, 2015.

[39] C. Spence, *Gastrophysics: The New Science of Eating*. Viking, 2017.

[40] M. Spanio, V. Frezzato, and A. Rodà, "Multimodal dataset normalization and perceptual validation for music-taste correspondences," 2026. [Online]. Available: https://arxiv.org/abs/2604.10632

[41] M. Spanio and A. Rodà, "Taste-aware music retrieval from audio embeddings," in *Proceedings of the 23rd IEEE International Conference on Content-Based Multimedia Indexing (CBMI)*. Toulouse, France: IEEE, Oct. 2026, accepted for publication.

[42] B. Elizalde, S. Deshmukh, M. A. Ismail, and H. Wang, "Clap learning audio concepts from natural language supervision," in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, June 2023, pp. 1–5.

[43] B. Elizalde, S. Deshmukh, and H. Wang, "Natural language supervision for general-purpose audio representations," in *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2024, pp. 336–340.

[44] K. Kilgour, M. Zuluaga, D. Roblek, and M. Sharifi, "Fréchet Audio Distance: A Reference-Free Metric for Evaluating Music Enhancement Algorithms," in *Interspeech 2019*, 2019, pp. 2350–2354.

[45] R. A. Bradley and M. E. Terry, "Rank analysis of incomplete block designs: I. the method of paired comparisons," *Biometrika*, vol. 39, no. 3/4, pp. 324–345, 1952.