

# 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 need controllable generation on commodity and embedded hardware rather than on 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 GPUs, CPU-only machines and a Raspberry Pi 5, without Python or a deep-learning framework. Our main contribution is a study of *quantization*, running the model at lower precision to fit tight memory budgets. We measure its quality cost against the variation between random seeds, on three checks from two independent model families: prompt adherence and distributional quality from a text–audio embedding model, and taste preservation from a taste regressor. Eight-bit precision stays within seed-to-seed noise on all three checks while sharply cutting memory, and with eight-bit arithmetic it is the fastest GPU mode. Four-bit measurably exceeds the noise, most on prompt adherence, but fits the medium model, a 1.45-billion-parameter denoiser, on a 4 GB Pi. Against the official implementation, *aria* starts about eight times faster from a fresh process and holds a smaller GPU footprint. At matched precision a compiled reference generates 10 s clips faster, but *aria*’s 8-bit mode overtakes it; long-form generation on the smaller model is the one case where the reference stays ahead. Because the runtime owns every tensor, it also exposes activation steering at negligible cost: a case study generating music with taste associations (*sonic seasoning*) shows genuine but bounded control on three of five attributes.

**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 about quality, but also about cold-start latency, predictable memory use, portability across commodity CPUs and GPUs, and keeping a model resident near the user. Open-weight music models are now good enough for such systems, yet their reference implementations still assume a Python/PyTorch serving stack and a discrete GPU, a mismatch for Internet-of-Sounds applications, where semantic audio runs as a local or edge service rather than a datacenter job.

Local-inference projects such as llama.cpp [2] and Dwarf-Star [3] show that, once the serving stack is gone, models treated as cloud-only run on hardware people already own. Music generation is a good candidate for that shift: open

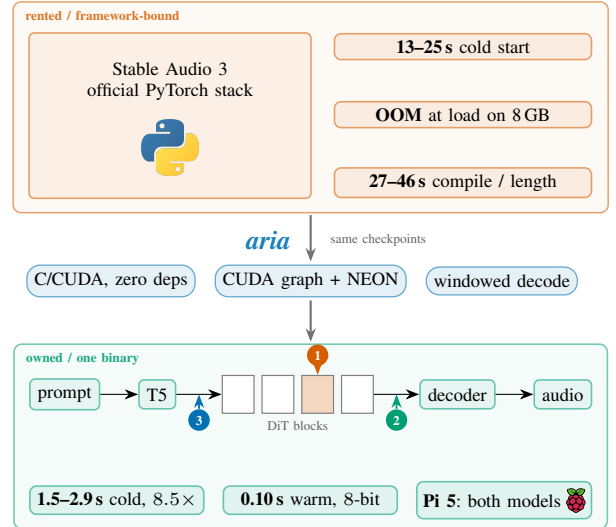


Fig. 1. With *aria* the same Stable Audio 3 checkpoints move from a framework-bound serving stack (top) to one dependency-free binary on owned hardware (bottom). Steering is injected at 1 the DiT residual stream, 2 the latent, or 3 the text conditioning.

music models have roughly 1–10 B parameters, far fewer than frontier language models, and diffusion inference scales with denoising steps rather than with long token sequences. The two leading open-source systems, ACE-Step 1.5 [4] and Stable Audio [5], are both diffusion transformers (DiTs). We take Stable Audio 3 (SA3) as the target and ask how much of its deployment cost belongs to the model and how much to the framework around it.

We answer with *aria*, a dependency-free C/CUDA runtime that runs SA3’s full text-to-music pipeline, for both model variants, on commodity GPUs and CPU-only hardware (Figure 1). It stores weights from half precision down to 8- or 4-bit integers, runs the arithmetic in 8-bit where that pays off, and bounds memory for long-form generation. On a Raspberry Pi 5 the small model generates a 10 s stereo clip at a 1.9 GB full-precision peak, which 8-bit weights cut to 0.84 GB; the medium model, whose denoiser alone has 1.45 B parameters, runs at 4-bit within 3.0 GB. Owning every intermediate tensor also makes activation steering a runtime primitive, injected into the generation graph at negligible cost, rather than a Python-side patch or a retrained adapter. Our case study is

sonic seasoning, taste-conditioned music generation [6]: taste is hard to put into words but externally measurable, a good test of whether a learned steering target tracks semantic change or merely responds to degradation.

Neither low-bit quantization nor activation steering is our invention [7]–[11]; we claim their *integration* into a framework-free runtime for a latent-diffusion music model, and measure what it costs in quality. Our contributions are: (i) **the aria runtime**, running SA3’s full pipeline on GPUs, CPU-only systems and embedded-class hardware, with a matched efficiency study against the official implementation covering warm latency, cold start, peak memory and CPU-only generation; (ii) a **deployment characterization of quantization**, from half precision down to 4-bit weights plus an 8-bit-arithmetic mode, releasing the full-precision weights once compressed and gating each precision on three checks from two independent models: 8-bit stays within re-seed noise on all three, while 4-bit exceeds it but puts the medium model on a 4 GB Pi; (iii) an **in-graph realization of activation steering** with no measurable overhead, bit-exact disable behaviour and dose–response parity with the reference implementation; and (iv) a **bounded semantic-control case study** applying sonic seasoning to a hard-to-lexicalize attribute under a multi-oracle protocol.

## II. RELATED WORK

### A. Generative audio models

Text-to-music systems are either autoregressive transformers over discrete acoustic tokens or latent-diffusion models that denoise in the latent space of a neural audio autoencoder. SA3 [5] couples a SAME semantic-acoustic autoencoder [12] to a DiT denoiser [13], conditioned on text via T5Gemma [14] cross-attention and trained with flow matching [15]; activation steering reads and writes the DiT residual stream of per-block hidden states. The shift toward live generation, as in Magenta/Lyria RealTime [16], also favours warm resident runtimes over cold per-request processes.

### B. Generative models on edge devices

State-of-the-art generative audio increasingly runs behind closed APIs such as Suno<sup>1</sup> and Udio<sup>2</sup>, while open-weight reference stacks are typically heavyweight Python pipelines that assume a discrete GPU; community CPU ports of SA3 remain tied to PyTorch [17]. The ggml lineage (llama.cpp [2], whisper.cpp [18], DwarfStar [3], stable-diffusion.cpp [19]) shows that a self-contained C/C++ program with memory-mapped weights, low-bit quantization, and hand-written vector and GPU kernels [20] can deliver interactive inference on consumer hardware with near-zero cold start. Portability-first engines [21]–[24] instead trade model-specific kernels for generic graph execution. Neither approach has reached latent-diffusion music: to our knowledge no dependency-free C runtime exists for a state-of-the-art music model.

<sup>1</sup><https://suno.com>, accessed 2026-07-04.

<sup>2</sup><https://www.udio.com>, accessed 2026-07-04.

### C. Quantization for on-device diffusion

Post-training quantization (PTQ) compresses LLMs to 8-bit weights *and* activations without retraining [7], diffusion-specific PTQ pushes image models to 4-bit weights [8], and PTQ has recently reached audio diffusion transformers [9]. These works tune the quantizer for a fixed fidelity target; we address deployment instead, treating precision as a runtime axis with hardware-specific 8-bit kernels and measuring each precision’s quality cost against fp16 rather than fixing a budget.

### D. Steering techniques

A frozen generative model can be controlled by editing its weights or by intervening on its activations at inference time. The activation route rests on the linear representation hypothesis: interpretable attributes are approximately linear directions in hidden space, so adding one vector to the residual stream shifts an attribute and leaves the rest of the computation intact. Activation addition [10] and representation engineering [11] extract such directions from contrastive examples, and Tan et al. [25] characterize when they generalize. The paradigm is now well represented in audio and music [26]–[31]; we follow the injection formulation of Camporese [26]. The weight-based family (LoRA [32], ControlNet [33], concept sliders [34]) gives strong control at the cost of training, per-attribute data and a growing parameter footprint; LoRA, the de-facto standard, is our training-based comparator. Our case study targets the basic tastes: “sonic seasoning” reports crossmodal associations between tastes and acoustic parameters [35], [36] stable enough to shift the perceived taste of food [37], [38], later synthesized as gastrophysics [39].

## III. METHODOLOGY

### A. The aria runtime

*aria* brings the dependency-free native-runtime approach of Section II to audio latent diffusion. It targets the settings where the official SA3 PyTorch stack is least practical, interactive generation from a resident service [16] and budget or CPU-only inference, in which framework startup, GPU-context setup and a multi-second cold start dominate short generations. It is a single-purpose engine in about 8.9k lines of C and CUDA (7.3k excluding blank and comment lines), with no linear-algebra library, deep-learning framework or other third-party dependency beyond the C math library and a threading runtime.

A small dispatch layer holds the whole pipeline: a text tokenizer, the T5Gemma text encoder [14], the DiT that denoises the audio latent over eight sampling steps, and the autoencoder [12] (a 256-dimensional latent at  $\sim 10.7$  Hz); a new architecture costs one file and a registry entry rather than a kernel rewrite. Parameter counts refer to the denoiser, which is what quantization acts on: 0.46 B (small-music) and 1.45 B (medium), against 0.85 B for medium’s autoencoder and 0.59 B for the shared text encoder, a  $\approx 2.9$  B pipeline. Weights are mapped from disk in half precision, with full precision kept only where fidelity is sensitive.

TABLE I

PRECISION LADDER (SMALL-MUSIC). TOP: DEPLOYMENT COST (MB; VRAM IS PEAK OVER IDLE, AS IN TABLE II; PI MEMORY IS PEAK RESIDENT SET SIZE, RSS, ON AN ACTIVELY COOLED 4 GB PI 5). BOTTOM: FIDELITY VS. FP16 ON THE CHECKS OF SECTION IV-A, WHOSE RE-SEED FLOORS ARE  $\Delta\text{CLAP} \pm 0.004$  (RAW COSINE),  $\text{FD}_{\text{CLAP}} 44.0$  AND  $\Delta\text{TASTE} 0.153$ .

|                           | fp32 | fp16 | q8     | W8A8        | q4     |
|---------------------------|------|------|--------|-------------|--------|
| GPU (s)                   | –    | 0.13 | 0.20   | <b>0.10</b> | 0.22   |
| VRAM                      | –    | 1345 | 1025   | 1025        | 801    |
| Pimem                     | 1912 | 1198 | 836    | 836         | 776    |
| $\Delta\text{CLAP}$       | –    | ref  | +0.002 | –0.001      | –0.020 |
| $\text{FD}_{\text{CLAP}}$ | –    | ref  | 18.6   | 17.7        | 125.8  |
| $\Delta\text{taste}$      | –    | ref  | 0.035  | 0.044       | 0.199  |

The same operations run 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. The GPU path uses half-precision tensor-core matrix multiplies and attention, including a banded decoder kernel, and replays the per-step denoising loop as one captured 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), so long-form audio streams even on a Raspberry Pi 5. A resident batch mode and an HTTP server keep the model loaded across jobs, cutting per-job cost  $3.7\times$  against a one-shot invocation (0.36 vs. 1.32 s).

a) *Steering interface*: *aria* loads precomputed direction vectors and applies them at three points, the transformer’s residual stream, the compact audio latent and the text conditioning, either adding the direction or projecting onto it, optionally within a chosen span of sampling steps. This reproduces the reference intervention exactly ( $\delta = \alpha\|h\|\hat{d}$ ; Section III-B). On the GPU the steering step sits inside the captured graph and reads a strength refreshed each step (zero outside the 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.

b) *Quantization*: The transformer’s weights can be stored in half precision or as 8- or 4-bit integers (q8, q4). q8 is per-output-row symmetric int8,  $s_n = \max_k |W_{nk}|/127$ ; q4 is per-block asymmetric int4 with a zero point (blocks of 32,  $s = (\max - \min)/15$ ,  $W \approx \min + qs$ ), which tracks skewed weight blocks far better than symmetric int4. Both derive from the weights alone, so no calibration set is needed. An optional mode also runs the arithmetic in 8-bit (W8A8), reusing the packed q8 weights without dequantization: activations are quantized *dynamically, per token*, by row absmax ( $s_m = \max_k |x_{mk}|/127$ , round-to-nearest, clamped to  $[-127, 127]$ ), the six per-step block matrix multiplies run int8 $\times$ int8 with exact int32 accumulation, and the result is rescaled in place,

$$y_{mn} = \text{acc}_{mn} s_m s_n + b_n. \quad (1)$$

Integer tensor cores carry this on the GPU and the dot-product instruction on ARM (with a widening path on x86), bit-identically. The cross-attention key/value projection, the

attention products and the autoencoder stay in half precision, being fidelity-sensitive. Because the runtime frees the full-precision source once a compressed copy is packed, lower precision *reduces* resident memory rather than adding to it. Each precision is gated against fp16 on three checks from two independent models, Contrastive Language–Audio Pretraining (CLAP) and a taste regressor (Section IV-A).

### B. Taste steering and sonic seasoning

This subsection exercises the steering interface end-to-end, testing the runtime’s control path rather than opening a separate line of work. Sonic seasoning associates musical features with basic tastes (consonant, high-pitched legato reads as *sweet*; dissonant, low music as *bitter*); we ask whether a frozen SA3 can be steered to *produce* audio carrying these associations along five axes, {*sweet, sour, bitter, salty, spicy*}.

1) *Direction and injection site*: Each DiT block writes its update additively, so its output lies in the same  $d_{\text{model}}$  space as its input. For axis  $i$  and block  $\ell$  we estimate a unit direction  $d_i^{(\ell)}$  from “low- $i$ ” to “high- $i$ ” by difference-in-means, the simplest estimator known to transfer [10], [11], [25], [26], over contrastive sets  $S_i^\pm$ :

$$d_i^{(\ell)} = \frac{\bar{h}_{i,\ell}^+ - \bar{h}_{i,\ell}^-}{\|\bar{h}_{i,\ell}^+ - \bar{h}_{i,\ell}^-\|_2}, \quad \bar{h}_{i,\ell}^\pm = \frac{1}{|S_i^\pm|} \sum_{x \in S_i^\pm} \frac{1}{T} \sum_{t=1}^T h_\ell(x)_t. \quad (2)$$

*Prompt-side* sets are caption pairs differing only in a taste descriptor; they are self-contained but encode only what the text encoder associates with that word. *Audio-side* sets ground the contrast in *norm-sonic-seasoning* [40], 377 music clips with per-clip basic-taste ratings: we feed the top- and bottom- $k$  clips on axis  $i$  to SA3 as an initial audio input, capturing the residuals it produces on audio humans heard as sweet or sour. Audio-side directions are more monotonic and more robust to over-steering, so we use them by default and keep prompt-side ones as a controlled comparison.

2) *Additive injection*: At inference we add a scaled copy of the direction to the chosen block’s output,

$$h \leftarrow h + \alpha \|\bar{h}\| d_i^{(\ell)}, \quad (3)$$

where  $\|\bar{h}\|$  is the mean baseline residual norm at  $\ell$ , so the single strength  $\alpha \geq 0$  is in units of the typical residual magnitude, comparable across blocks and model sizes. Setting  $\alpha = 0$  leaves the pass unchanged and gives the matched, seed-identical baseline. We sweep  $\alpha$  up to 1.0 because the response is non-monotonic.

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”). Both arms share extraction data, pipeline and evaluation.

4) *Multi-oracle evaluation*: The central risk is circularity: the quantity we optimize is also the one tempting to report as evidence. 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. 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, so we guard it with three checks over CLAP [42], [43] embeddings, which share no training signal with it: **CLAP similarity** to “X-tasting music” anchors, a semantic cross-check;  $\text{FD}_{\text{CLAP}}$ , a Fréchet distance [44] between CLAP embeddings reduced to 32 dimensions, a degradation detector; and **audio drift**,  $1 - \cos(\phi(x_\alpha), \phi(x_0))$ , the perturbation magnitude. We write  $\text{FD}_{\text{CLAP}}$ , not FAD: its values are comparable neither to the VGGish-based Fréchet Audio Distance nor across embedding spaces. The three checks are independent of the target but not of each other, so we read them as one guard rather than three votes.  $\text{FD}_{\text{CLAP}}$  and drift are measured against the unsteered  $\alpha=0$  clips: they say how far the intervention moved the audio, not its distance from real music, so only their behaviour relative to CLAP similarity is diagnostic.  $\Delta\text{CLAP}$  is in the logit-scaled similarity units msclap returns and  $\text{FD}_{\text{CLAP}}$  uses a per-run PCA basis, so neither is on the scale of the quantization study (Table I).

The operating point follows an explicit rule: among strengths with a significant target shift and  $\Delta\text{CLAP} > 0$ , take the  $\alpha$  maximizing  $\Delta\text{CLAP}$ , and claim an axis only if that maximum exceeds one similarity point. On Table III this admits exactly SWEET (+2.11 at  $\alpha=0.3$ ), SOUR (+1.77) and BITTER (+1.66, both at  $\alpha=0.1$ ), rejecting SALTY (+0.32) and SPICY (+0.25, at  $\alpha=0.15$ ) as CLAP-marginal. Where  $\Delta_i$  rises while CLAP falls and  $\text{FD}_{\text{CLAP}}$  or drift increase, we report degradation instead. Selection runs in two stages: a target-only per-layer scan nominates one candidate block per axis, and a dense  $\alpha$ -window at that block is then gated by all four oracles, so no operating point is chosen by the target alone.

#### IV. EXPERIMENTS AND RESULTS

##### A. Experimental setup

All conditions use 10s clips, loudness-normalized to  $-14\text{LUFS}$  (loudness units relative to full scale) before scoring, with seeds  $\{0, 1, 2\}$  and means reported. We steer two released configurations: small-music (fp32, 20 blocks,  $d_{\text{model}}=1024$ ) and medium (fp16, 24 blocks,  $d_{\text{model}}=1536$ ). Medium in fp32 exceeds the 8GB reference GPU, so we also re-run small-music in fp16: its two dose-responses coincide (pooled  $r=1.00$ , mean absolute error (MAE) 0.003 over all five axes), so medium-small differences reflect scale rather than precision. Per-point significance uses a two-sided Wilcoxon signed-rank test on the  $n=36$  paired deltas, Holm-corrected across the five axes.

a) *Steering protocol*: We compare prompt- and audio-side directions by dose-response monotonicity (Spearman  $\rho$ ) and over-steer robustness, holding block, grid and seeds fixed. To locate taste, a logistic probe finds the layers where the axes are linearly separable; we then steer every DiT block at  $\alpha = 0.15$  (5 prompts  $\times$  3 seeds) and rank blocks by the target alone. At each axis’s best small-music block 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 oracles; a model-size study repeats the **sweet** sweep on medium at its own re-scanned best layer. Three ablations reuse this panel on small-music: the *injection op*, adding projection, which amplifies ( $\beta>0$ ) or removes ( $\beta<0$ ) the direction component already present in each token; the *step window*, confining injection to the early (0–3) or late (4–7) denoising steps at matched strength; and the *site*, applying the same recipe to the compact audio latent and the pooled text embedding. The LoRA comparator is trained for 800 steps, dose-swept and scored with the same panel.

b) *Runtime fidelity and efficiency*: To confirm that the runtime preserves steering behaviour, we re-run the dense window through *aria* with the identical exported directions and  $\alpha$  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 an RTX 3070 (8GB) and, CPU-only, on its i9-10900KF host, against the official SA3 implementation (commit dedace19, 2026-07-01; torch 2.7.1, CUDA 12.6) in half precision, with its native 8-step sampler and no classifier-free guidance. We separate *warm* generation, with the model resident in-process on both sides (the per-request cost of a live or batched service), from *cold start*: process launch, weight load and one generation, what an edge endpoint pays on first use. Both runtimes are measured identically: one warm-up generation is discarded and the mean of the next eight is reported, and VRAM is the peak GPU memory over the idle baseline, sampled at 50 ms, so each side is charged for its CUDA context and allocator pool; memory is never sampled during a timing run, because polling the driver perturbs generation time. The repository disables half precision and transformer compilation by default; we report it both as shipped and compiled, because the two give opposite answers.

c) *Quantization fidelity*: Clips are 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. We use 24 genre-diverse prompts  $\times$  3 seeds (72 clips per precision) and three metrics from two independent models: *prompt adherence*, the change in each clip’s CLAP [43] similarity to its own prompt; *distributional quality*, the  $\text{FD}_{\text{CLAP}}$  of Section III-B between the precision’s clips and the fp16 set; and *taste preservation*, the distance between the 5-D wav2taste vectors of each clip and its fp16 counterpart. Re-seeding fp16 (seeds 3–5) gives each metric a noise floor, the change a re-seed alone induces; a precision is within noise below it. The two comparisons deliberately differ: a precision is scored against fp16 at the same seeds, so both sides share musical content and differ only in numerics, whereas the floor compares fp16 with itself across disjoint seeds. Scoring below the floor therefore means that the precision perturbs the output less than the seed does, the deployment question; it is not a claim that quantized audio is closer to fp16 than fp16 is to itself.

d) *Artifact availability*: The *aria* runtime (source and build) is released at <https://github.com/matteospanio/aria>, with the SA3 weights and the exported steering vectors on Hugging Face (<https://huggingface.co/matteospanio/aria>).

TABLE II

EFFICIENCY OF *aria* VS. THE OFFICIAL SA3 IMPLEMENTATION: TIME (S) AND VRAM (MB, PEAK OVER IDLE) FOR A STEERED 10 S CLIP, 8 STEPS, ON AN RTX 3070; CPU ROWS USE ITS 20-THREAD I9-10900KF. PROTOCOL IN SECTION IV-A. *aria* VARIES MORE RUN TO RUN THAN THE COMPILED BASELINE (SMALL-MUSIC S.D. 0.026 VS. 0.001 S).

| Configuration        | small-music  |             | medium       |             |
|----------------------|--------------|-------------|--------------|-------------|
|                      | s            | MB          | s            | MB          |
| aria, warm fp16      | 0.146        | <b>1348</b> | 0.364        | <b>4148</b> |
| aria, warm W8A8      | <b>0.100</b> | 1179        | <b>0.281</b> | 3221        |
| aria, fresh process  | <b>1.47</b>  | –           | <b>2.89</b>  | –           |
| aria, CPU-only warm  | 2.45         | –           | 9.65         | –           |
| SA3, warm as shipped | 0.396        | 1960        | 0.569        | 5612        |
| SA3, warm compiled   | <b>0.124</b> | –           | <b>0.291</b> | –           |
| SA3, fresh process   | 13.1         | –           | 24.5         | –           |

### B. Runtime benchmarks

Warm throughput depends on how the baseline is configured (Table II). As shipped, the reference is  $2.7\times$  (small-music) and  $1.6\times$  (medium) slower than *aria* on a steered 10 s clip; compiling its transformer reverses that at equal precision, leaving fp16 *aria*  $1.18\times$  and  $1.25\times$  slower. Equal precision is not the operative comparison for a deployment, though: the two runtimes need not spend the same bits for the same quality, and *aria*’s W8A8 mode beats the compiled reference by  $1.24\times$  and  $1.03\times$ , without compilation, while staying inside every re-seed floor of Table I. At matched fp16 the compiled reference is faster; at matched *output quality*, *aria* is.

Compilation is also tied to clip length: the first 10 s request costs 26.6 s to compile and the first 12 s request a further 45.8 s, after which PyTorch falls back to a dynamic-shape kernel that stops recompiling but runs slower (0.165 s) than fp16 *aria*. A fixed-length service should compile the reference; a varying-length one pays either the recompiles or the slower kernel.

The margins that do not depend on that choice are larger: *aria* starts  $8.5\text{--}8.9\times$  faster from a fresh process and holds a  $1.35\text{--}1.45\times$  smaller peak GPU footprint. It also runs without a GPU: after parallelizing and vectorizing the elementwise work between the matrix multiplies, a 10 s clip takes 2.45 s on small-music ( $\sim 4\times$  real time) and 9.65 s on medium, on the 20-thread CPU. Chunked sliding-window generation streams at  $\sim 4\times$  real time on the RTX 3070 ( $\sim 0.6\times$  CPU-only), and a six-step preset trims a further  $\sim 25\%$  off every transformer-bound path at a taste shift below the seed-to-seed noise floor.

Long-form generation separates the tiers. At 60 s on medium, a half-precision attention output stage and running the mapping convolution as a matrix multiply bring *aria* level with the reference at equal precision (1.30 vs. 1.31 s), and W8A8 takes it to 1.21 s,  $1.09\times$  ahead. On small-music the reference stays ahead (0.407 vs. 0.455 s at 8-bit,  $1.12\times$ ), the one configuration where *aria* still trails, owing to the DiT’s full  $O(S^2)$  self-attention against the reference’s fused kernel. These long-form figures are unconditional, excluding the text encoder on both sides, and compiling the reference does not help at this length (1.335 and 0.420 s), where the matrix multiplies dominate. On CPU the banded decoder renders 60 s

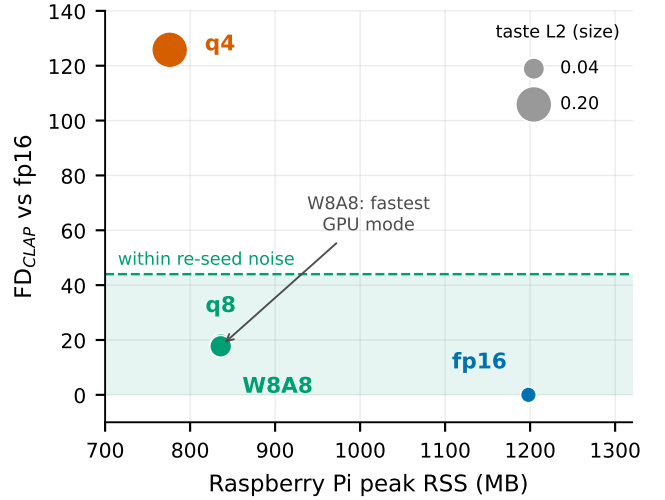


Fig. 2. Quantization in one space (small-music, GPU, fp16 decoder, so only DiT precision differs). Left is less Pi memory, up is  $FD_{CLAP}$  vs. the fp16 set, and marker area grows with the taste-vector L2. The shaded band is the  $FD_{CLAP}$  an fp16 re-seed induces.

of medium  $2.15\times$  faster (48.8 vs. 104.9 s). The reference’s medium loader briefly needs  $\sim 7.1$  GB, overflowing the 8 GB card, so it used a low-memory load path.

a) *Quantization is close to free at 8-bit, and enables the edge*: Table I and Figure 2 characterize precision as a deployment axis. Validating quantization with the oracle the steering study optimizes would be circular, so we lead with the two CLAP-based metrics and keep the wav2taste distance as a narrow corroborator. Eight-bit shows no measurable degradation: q8 and W8A8 both stay inside every re-seed floor while cutting GPU memory  $\sim 24\%$  against fp16 and Pi peak memory  $2.3\times$  against fp32, and W8A8 is the fastest GPU mode (0.10 s warm, on the integer tensor cores). Four-bit crosses every floor, most on prompt adherence at  $5\times$  its floor. It is still what runs the 1.45 B-denoiser medium model on a Raspberry Pi 5 within a 3.0 GB peak and 0.7 GB resident, so a 4 GB board suffices; we report it as a memory-fit result, the fidelity numbers above being small-music.

b) *Steering during streaming*: Because steering runs inside the graph on a warm model, it can be changed during a stream: with the SWEET strength ramped  $0 \rightarrow 0.5 \rightarrow 0$  across a 12-chunk stream, the per-chunk taste score tracks the schedule (Spearman  $\rho=0.78$ ,  $p=0.003$ ), lagging on the descent as the continuation inherits context. On the Raspberry Pi 5 a steered clip costs the same as an unsteered one (28.2 vs. 28.4 s), so semantic control stays fully on-device, though not in real time: the Pi needs  $\sim 28$  s per 10 s clip ( $\sim 0.35\times$  real time), and only the GPU path reaches real time. Back-to-back 10 s clips hold  $\sim 62^\circ\text{C}$  without throttling, but a multi-minute medium generation reaches the  $85^\circ\text{C}$  soft limit on a passively cooled board and is clocked down, costing 19% (319 vs. 267 s); a fan removes that, holding  $63^\circ\text{C}$  throughout.

### C. Steering results

*Extraction source*. On small-music, the prompt-derived SWEET direction yields an inverted-U response, peaking at

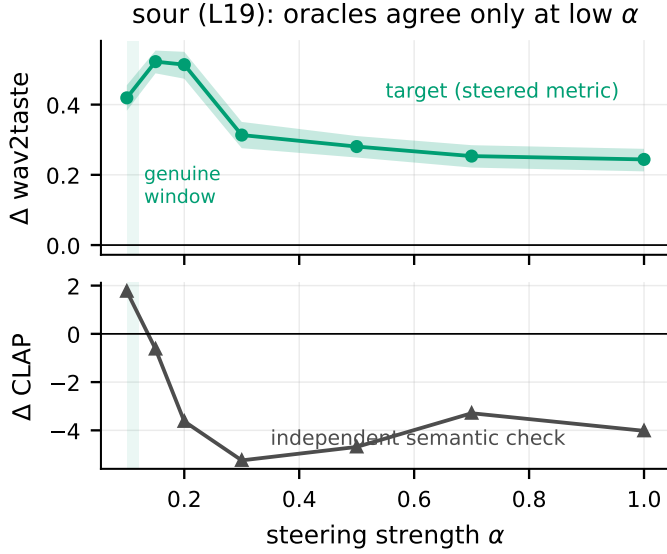


Fig. 3. Degradation demonstration (SOUR, small-music, L19). *Top*: the wav2taste target (95% bootstrap CI) keeps climbing past the shaded genuine window. *Bottom*: the independent CLAP check turns negative there.  $FD_{CLAP}$  (Table III) rises monotonically throughout.

TABLE III

DENSE-WINDOW MULTI-ORACLE STEERING (SMALL-MUSIC, FP32;  $n=36$  PER POINT), EACH AXIS AT ITS WAV2TASTE-BEST LAYER. UNITS, THE  $\alpha=0$  REFERENCE FOR  $FD_{CLAP}$  AND DRIFT, AND THE SELECTION RULE ARE IN SECTION III-B; \* MARKS THE SELECTED OPERATING POINTS AND **BOLD** A NEGATIVE  $\Delta CLAP$ .

| Axis                                                     | $\alpha$ | $\Delta_{w2t}$ | $\Delta CLAP$ | $FD_{CLAP}$ | drift |
|----------------------------------------------------------|----------|----------------|---------------|-------------|-------|
| <i>sweet (L16): full sweep</i>                           |          |                |               |             |       |
| 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  |
|                                                          | 1.0      | -0.025         | <b>-3.88</b>  | 1257        | 0.72  |
| <i>intense (L19): genuine at <math>\alpha=0.1</math></i> |          |                |               |             |       |
| sour                                                     | 0.1*     | +0.419         | +1.77         | 561         | 0.40  |
|                                                          | 0.15     | +0.522         | <b>-0.63</b>  | 985         | 0.62  |
| bitter                                                   | 0.1*     | +0.314         | +1.66         | 836         | 0.54  |
|                                                          | 0.15     | +0.360         | <b>-1.15</b>  | 1222        | 0.73  |
| salty                                                    | 0.1      | +0.209         | +0.32         | 576         | 0.41  |
|                                                          | 0.15     | +0.191         | <b>-0.98</b>  | 885         | 0.57  |
| spicy                                                    | 0.1      | +0.192         | <b>-0.18</b>  | 301         | 0.26  |
|                                                          | 0.15     | +0.267         | +0.25         | 574         | 0.42  |

$\Delta_{w2t}=+0.05$  ( $\alpha=0.1$ ) and collapsing to  $-0.08$  at  $\alpha=0.2$ ; the audio-derived direction reaches the same peak but stays monotonic (Spearman  $\rho=+0.94$ ) and positive at  $\alpha=0.2$ , so we use audio-side directions throughout.

*a) A narrow clean window:* Our central result is that wav2taste alone is not a reliable guide. Sweeping  $\alpha$  for SOUR at its best layer L19 (Figure 3, Table III), the oracles agree only at low strength. At  $\alpha=0.1$  the shift is large ( $\Delta_{w2t}=+0.42$ ,  $p=2.9 \times 10^{-11}$ ,  $d=3.71$ ) and CLAP is positive. Beyond it, wav2taste climbs to its maximum at  $\alpha=0.15$  while CLAP turns negative and  $FD_{CLAP}$  jumps; by  $\alpha=1.0$  the audio has collapsed into noise, which wav2taste, weakest on sour, misreads as intense taste. The target’s maximum thus lies

TABLE IV

aria REPRODUCES THE SA3/PYTORCH STEERING DOSE-RESPONSE: PER-AXIS PEARSON  $r$  AND MAE OVER THE SHARED PER-(AXIS,  $\alpha$ ) POINTS ( $n=35$  PER MODEL).

|        |          | sweet | sour  | bitter | salty | spicy | pooled       |
|--------|----------|-------|-------|--------|-------|-------|--------------|
| small  | w2t $r$  | 0.984 | 0.962 | 0.979  | 0.943 | 0.666 | <b>0.953</b> |
|        | w2t MAE  | 0.035 | 0.040 | 0.022  | 0.047 | 0.067 | <b>0.042</b> |
|        | CLAP $r$ | 0.979 | 0.985 | 0.973  | 0.870 | 0.545 | <b>0.859</b> |
|        | CLAP MAE | 1.02  | 1.42  | 1.57   | 2.00  | 4.29  | <b>2.06</b>  |
| medium | w2t $r$  | 0.976 | 0.949 | 0.916  | 0.833 | 0.614 | <b>0.916</b> |
|        | w2t MAE  | 0.033 | 0.040 | 0.037  | 0.052 | 0.028 | <b>0.038</b> |
|        | CLAP $r$ | 0.975 | 0.921 | 0.971  | 0.934 | 0.956 | <b>0.946</b> |
|        | CLAP MAE | 0.46  | 1.06  | 1.40   | 1.26  | 0.88  | <b>1.01</b>  |

inside the degradation regime. SWEET, SOUR and BITTER steer genuinely inside the clean window, with CLAP confirming, while SALTY and SPICY stay weak even when clean; we claim usable control for the first three axes only.

*b) Selection is separated from evaluation:* Layer and strength are chosen on the sweep prompts by the rule of Section III-B, then frozen and re-tested on ten prompts used for neither choice. Both replicate (SWEET +0.164, SOUR +0.508;  $n=30$  per axis, Holm  $p=3.7 \times 10^{-9}$ ); the held-out set covers SWEET and SOUR only, so BITTER rests on the selection sweep alone. 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  $\alpha=0.3$ ), so the window is neither a layer-selection nor a text-anchoring artefact.

*c) 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  $\alpha=0.15$  shows SWEET peaking mid-late at L16 and collapsing at the final blocks, while the four intense tastes rank the *final* block first (medium behaves alike, peaking at L20–21). The 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  $\Delta_{w2t}=+0.119$  at  $\alpha=0.3$  ( $p=3.2 \times 10^{-9}$ ,  $d=1.20$ ) with CLAP +2.11.

*d) Model scale:* The larger backbone widens the clean window instead of merely inflating the target. On SWEET, medium reaches a higher clean peak ( $\Delta_{w2t}=+0.143$  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  $\alpha=0.1$  it also degrades less on the intense tastes, with lower  $FD_{CLAP}$  across axes and CLAP still positive.

*e) 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 and SPICY noisiest on both. CLAP agrees on medium ( $r=0.95$ ) but less on small ( $r=0.86$ ), a numerical offset (*aria* runs in half precision where the small reference runs in full) that vanishes on medium. Both the genuine and the degradation regimes carry over.

TABLE V  
INJECTION-OP, STEP-WINDOW, STEERING-SITE AND LoRA ABLATIONS  
(SMALL-MUSIC, FP32;  $n=36$  PER POINT). COLUMNS AND UNITS AS IN  
TABLE III.

| Axis                                                       | Method / site     | Op-point      | $\Delta w_{2t}$ | $\Delta \text{CLAP}$ | $\text{FD}_{\text{CLAP}}$ |
|------------------------------------------------------------|-------------------|---------------|-----------------|----------------------|---------------------------|
| <i>Injection op (DiT residual)</i>                         |                   |               |                 |                      |                           |
| sour                                                       | additive          | $\alpha=0.1$  | +0.419          | +1.77                | 561                       |
|                                                            | additive          | $\alpha=0.15$ | +0.522          | <b>-0.63</b>         | 985                       |
|                                                            | projection        | $\beta=1$     | +0.492          | +0.60                | 544                       |
| sweet                                                      | additive          | $\alpha=0.3$  | +0.119          | +2.11                | 515                       |
| <i>Denoise-step window (DiT residual)</i>                  |                   |               |                 |                      |                           |
| sour                                                       | early (steps 0–3) | $\alpha=0.2$  | +0.530          | <b>-4.26</b>         | 1294                      |
|                                                            | late (steps 4–7)  | $\alpha=0.2$  | +0.515          | +0.80                | 565                       |
| <i>Steering site</i>                                       |                   |               |                 |                      |                           |
| 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          | <b>-3.04</b>         | 935                       |
| <i>Method: LoRA (compare with the steering rows above)</i> |                   |               |                 |                      |                           |
| sour                                                       | LoRA              | $\alpha=1.0$  | +0.341          | <b>-0.92</b>         | 767                       |
| sweet                                                      | LoRA              | $\alpha=0.5$  | +0.032          | <b>-0.10</b>         | 171                       |

f) *Ablations*: Three ablations run through the runtime’s batch mode, which the PyTorch injector cannot do (Table V); every operator, window and site point cited below is significant at  $p < 10^{-10}$ . *Operator*: projection amplification matches additive steering’s peak SOUR shift at much lower  $\text{FD}_{\text{CLAP}}$  with CLAP still positive, but does nothing for SWEET, whose residual component is too small to amplify; steering *away* from the direction barely moves either axis, so the operators are complementary across axes. *Window*: confining injection to the late steps (4–7) roughly halves the damage at a matched taste shift, whereas the early steps collapse CLAP; early steps appear to set global structure and late steps to shape the timbre both oracles read. *Site*: the compact latent steers SOUR cleanly, at a better quality-per-shift point than the residual, but inverts SWEET. The text-embedding site fails both axes at every scale tested (0.5–4): a mean shift there seems to move the conditioning off the data manifold rather than encode “more taste”, the prompt-side extraction failure reappearing at the embedding level. SOUR, an overtly timbral attribute, steers at latent and residual sites alike, while SWEET responds only to mid-late residual injection.

g) *Steering versus LoRA*: Under the same panel, the trigger-token LoRA (rank 8, 800 steps) never reaches a CLAP-positive operating point (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 rank-32 adapter trained  $3\times$  longer lands in the same place. The cost is one-sided too: 5.15 M parameters and 10.4 MB per axis after  $\sim 5$  min of training, against a 4–6 KB vector that trains nothing and can be removed at serve time. For this lexically ill-defined attribute, parameter updates add no control over a simple steering vector.

## V. DISCUSSION

### A. Where a native runtime actually pays

At matched fp16 precision and a fixed clip length, a compiled PyTorch baseline is faster than *aria*, and we do

not claim otherwise. *aria*’s advantage comes from owning the numerics, which lets it spend fewer bits for the same quality, and from removing framework startup, GPU-context setup and per-length compilation, the costs that dominate short generations and varying-length services. Interactive and embedded settings therefore gain most from a resident native runtime, and quantization costs them no speed: 8-bit arithmetic is also the fastest GPU mode.

### B. Learned oracles require independent checks

Tune an intervention to maximize a learned metric and that metric stops being a faithful indicator once the intervention grows too aggressive: the target keeps rising while independent checks collapse. Layer selection has the same problem, since the target can mistake degradation for success. Both the strength and the site therefore need supervision from signals that share no training path with the target, a caution that extends to any representation-editing evaluation reporting the optimized quantity as its own evidence.

### C. Scope and limitations

The systems contributions rest on direct measurement. The steering study is a bounded case study: it intervenes on one model family and is judged by learned or distributional oracles, so we claim genuine control only for the three attributes where independent checks confirm it, and treat the other two as a negative result. Those operating points replicate on held-out prompts, survive an audio-anchored re-test, and hold across a precision change and the runtime port, while a stronger LoRA adapter fares no better (Section IV-C).

## VI. CONCLUSION

We presented *aria*, a dependency-free C/CUDA runtime that deploys Stable Audio 3 without a datacenter-oriented serving stack. Against the official implementation it starts about  $8.5\times$  faster from a fresh process, holds a  $1.35\text{--}1.45\times$  smaller GPU footprint and runs CPU-only at  $\sim 4\times$  real time; a compiled reference is  $1.18\text{--}1.25\times$  faster warm at matched fp16 precision, but *aria*’s 8-bit mode overtakes it by  $1.03\text{--}1.24\times$  at a quality that re-seed noise cannot separate from fp16. Treated as a deployment axis, precision behaves well: 8-bit stays within re-seed noise on all three checks and runs fastest on the GPU, and 4-bit puts the 1.45 B-denoiser medium model on a 4 GB Pi. The bit-exact steering interface reproduces the reference dose–response, and the sonic-seasoning case study shows taste control that is real but confined to a narrow operating window and to three of five attributes. Semantic control should be evaluated with independent quality checks, not with the optimized metric alone.

### A. Outlook

The remaining small-tier long-form gap comes from the transformer’s full self-attention, whose weights we measure to be too spread out for a banded approximation; a fused attention kernel is the open lever. A persistent multi-client server mode would also support latent-domain streaming continuation [16].

The main open validation is perceptual: a study with human participants should test whether the steered clips shift perceived taste as the automatic oracles indicate.

#### DISCLOSURE OF AI USE

We used large language model assistants when drafting and revising parts of this manuscript and as coding assistants while implementing aria. They were not used to produce or to interpret any measurement. The authors derived and checked the adjoints, designed and ran every experiment, verified the code and the numbers reported here, and take full responsibility for them.

#### 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 *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, April 2025, pp. 1–5.
- [2] G. Gerganov and contributors, “llama.cpp: LLM inference in C/C++,” <https://github.com/ggml-org/llama.cpp>, 2023.
- [3] S. Sanfilippo and contributors, “DwarfStar,” <https://github.com/antirez/ds4>, 2026.
- [4] J. Gong *et al.*, “ACE-Step 1.5: Pushing the boundaries of open-source music generation,” arXiv:2602.00744, 2026.
- [5] Z. Evans *et al.*, “Stable Audio 3,” arXiv:2605.17991, 2026.
- [6] M. Spanio, M. Zampini, A. Rodà, and F. Pierucci, “A multimodal symphony: integrating taste and sound through generative AI,” *Frontiers in Computer Science*, vol. 7, 2025.
- [7] 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 *Proc. Int. Conf. Mach. Learn. (ICML)*, 2023.
- [8] X. Li *et al.*, “Q-Diffusion: Quantizing diffusion models,” in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, 2023.
- [9] T. Khandelwal and M. Fuentes, “Post-training quantization for audio diffusion transformers,” arXiv:2510.00313, 2025.
- [10] A. M. Turner, L. Thiergart, D. Udell, G. Leech, U. Mini, and M. MacDiarmid, “Activation addition: Steering language models without optimization,” arXiv:2308.10248, 2023.
- [11] A. Zou *et al.*, “Representation engineering: A top-down approach to AI transparency,” arXiv:2310.01405, 2025.
- [12] J. D. Parker *et al.*, “SAME: A semantically-aligned music autoencoder,” arXiv:2605.18613, 2026.
- [13] W. Peebles and S. Xie, “Scalable diffusion models with transformers,” in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct 2023, pp. 4172–4182.
- [14] B. Zhang, F. Moiseev, J. Ainslie, P. Suganthan, M. Ma *et al.*, “Encoder-decoder Gemma: Improving the quality-efficiency trade-off via adaptation,” arXiv:2504.06225, 2025.
- [15] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le, “Flow matching for generative modeling,” in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2023. [Online]. Available: <https://mlanthology.org/ficlr/2023/lipman2023iclr-flow/>
- [16] A. Caillon *et al.*, “Live music models,” in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), Creative AI Track*, 2025. [Online]. Available: <https://openreview.net/forum?id=SB3XBpGvpl>
- [17] innermost47, “OBSIDIAN-Neural: Stable Audio 3 Medium running locally on CPU,” <https://github.com/innermost47/ai-dj>, 2026, local edition;  $\approx 11$  s/generation on a laptop CPU.
- [18] G. Gerganov and contributors, “whisper.cpp: Port of OpenAI’s Whisper model in C/C++,” <https://github.com/ggml-org/whisper.cpp>, 2022.
- [19] J. Ng *et al.*, “Open-source acceleration of stable-diffusion.cpp deployable on all devices,” arXiv:2412.05781, 2025.
- [20] D. Gope, D. Mansell, D. Loh, and I. Bratt, “Highly optimized kernels and fine-grained codebooks for LLM inference on Arm CPUs,” arXiv:2501.00032, 2024.
- [21] ONNX Runtime developers, “ONNX Runtime,” <https://onnxruntime.ai/>, 2021.
- [22] M. Nachin *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>
- [23] 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>
- [24] NVIDIA Corporation, “TensorRT,” <https://developer.nvidia.com/tensorrt>, 2018.
- [25] D. Tan *et al.*, “Analysing the generalisation and reliability of steering vectors,” in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 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 *et al.*, “Activation patching for interpretable steering in music generation,” arXiv:2504.04479, 2025.
- [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<sup>4</sup>-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 *Proc. Int. Conf. Learn. Represent. (ICLR)*, 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=jVSIJk5qNA>
- [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 *et al.*, “LoRA: Low-rank adaptation of large language models,” in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2022. [Online]. Available: <https://openreview.net/forum?id=nZeVKeeFY9>
- [33] L. Zhang, A. Rao, and M. Agrawala, “Adding conditional control to text-to-image diffusion models,” in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (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 *Proc. Eur. Conf. Comput. Vis. (ECCV 2024)*, 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,” in *Proc. 4th Workshop Artif. Intell. Cultural Heritage (IAI4CH), co-located with AIXIA*, 2026, arXiv:2604.10632.
- [41] M. Spanio and A. Rodà, “Taste-aware music retrieval from audio embeddings,” in *Proc. IEEE Int. Conf. Content-Based Multimedia Indexing (CBMI)*, 2026, arXiv:2607.03296.
- [42] B. Elizalde, S. Deshmukh, M. A. Ismail, and H. Wang, “CLAP learning audio concepts from natural language supervision,” in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, June 2023, pp. 1–5.
- [43] B. Elizalde, S. Deshmukh, and H. Wang, “Natural language supervision for general-purpose audio representations,” in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (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.