RIME: Enabling Large-Scale Agentic Music Post-Production

Noah Schaffer [1] **Nikhil Singh** [1]

Abstract

We formalize the task of *agentic post-production*, wherein individual aspects of a song are targeted, refined, and combined into a final track. Despite the promise of music generation models for one-shot output, such fine-grained iterative refinement workflows are a complementary problem and largely out of their reach. We argue the bottleneck is data: existing corpora do not reflect how realistic post-production chains map onto the vocabulary musicians and engineers actually use. We introduce the **Rule-based Instructions for Music Editing (RIME)** framework, which generates realistic paired edit-instruction data from any baseline music dataset grounded in canonical methods, design patterns, and constraints derived from real production workflows. RIME leverages POEMS, a new toolkit that combines stem separation, mixing, and common studio effects for use by multimodal agents. We use POEMS and RIME to generate 3,000 pairs of edit instructions and ground truth audio, and use this data to evaluate existing multimodal LLMs as agents on this task, showing persistent challenges in current models' post-production capabilities.

1. Introduction

There is a pervasive loop in the studio. When a producer listens to a recording they're working on, they will often hear something wrong, or something that could be better. They may isolate the offending stem, apply a chain of effects, blend it back in, and listen again. This loop has its own peculiar language. When a musician asks for something "warmer" or "bigger," a seasoned engineer translates it into a specific recipe (e.g. a high-shelf cut at 8 kHz, parallel compression on the drum bus, and a tempo-synced reverb send on the vocal) and runs it. The translation between what music *feels* like and the sequence of operations that produces that feeling is not, in fact, magic. It is a dense and surprisingly consistent vocabulary, built up over decades of studio practice. Today's AI music systems are typically not part of this loop. Yet, we have good reason to desire them to be: it admits an iterative, interactive workflow that is musician-centered at its core.

Altogether, no existing benchmarks match this loop, across triplet-based editing (Wang et al., 2023; Liang et al., 2025; Jia et al., 2025; Ellis et al., 2025; Han et al., 2024), Fx-chaining/removal (Doh et al., 2026; Imort et al., 2022; Rice et al., 2023), or inference-time audio editing (Liang et al.; Tsai et al., 2024; Chu et al., 2025; Zhang et al., 2024; Manor & Michaeli, 2024; Yang et al., 2026; Lan et al., 2026). For this, such a benchmark would need to (i) introduce both intentional modifications and degradations to audio, (ii) apply edits to individual stems and introduce them back into a mix, (iii) be able to handle instructions at varying abstraction levels, and (iv) support an iterative, listen-and-tweak workflow.

We introduce the **Rule-based Instructions for Music Editing (RIME)** framework, which generates large-scale music editing data that follows a process closer to studio music production. RIME starts from individual mixes, creates edit "recipes" using a large vocabulary of common studio engineering rules, constraints, and design patterns, and executes a series of tool calls to isolate sources, apply edits, and create a final mix. Additionally, RIME creates prompts at varying levels of abstraction to describe the edit chain for a given recipe, mimicking the variation in how precisely these intents are described in the real world. To create editing data, RIME leverages a newly built suite of tools that allow for **p**itch, **o**rnamentation (various effects), **e**qualization, **m**ixing, and source **s**eparation (POEMS) edits.

Overall, this work contributes:

1. We formalize the **agentic post-production** task
2. We introduce RIME, **a scalable, reusable data generation framework** for this task
3. To support this, we introduce POEMS, **an audio post-production toolkit**
4. As part of this, we **design and evaluate zero-shot agent baselines**, surfacing major limitations to current-generation models' abilities on this task

[1]Department of Computer Science, Dartmouth College, Hanover NH. Correspondence to: Noah Schaffer <noah.schaffer.gr@dartmouth.edu>.

Proceedings of the 43^{rd} International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026.

2. Methods

2.1. The POEMS Toolkit

Pitch We focus on pitch correction (autotuning) and harmony generation. Autotuning quantizes the fundamental frequency (f_0) contour of a vocal track to the nearest pitch within a target scale, specified by a key and mode.

Ornamentation and Equalization We organize spectral and temporal effects into time-based (e.g. reverb, delay), modulation (e.g. phaser, chorus), dynamics (e.g. compressor, limiter, waveshaper), and equalization (e.g. peak/bell, shelf, and hi/lo-pass filters). We use Pedalboard (Sobot, 2021) to implement these.

Separation and Mixing For source separation, we use the 6-stem variant of Demucs (Rouard et al., 2023), which restricts the set of editable stems in POEMS to vocals, drums, bass, piano, and guitar. Mixing then covers the operations that combine and balance stems. Within an edit graph, we re-combine a processed stem with its residual via linear summation in the time domain.

MCP Tool-Calling We expose the POEMS toolkit via Model Context Protocol (MCP) (Anthropic, 2024). GPU-bound tools (key detection, source separation) are served behind a FIFO queue that serializes device access. The remaining CPU-bound effects are stateless and can be invoked concurrently. This lets a tool-calling agent issue a sequence of MCP calls (e.g. separating the input mix, applying processing to one or more stems, and recombining them with the residual) to produce an edited mix end-to-end.

2.2. RIME

Recipes and Motifs A *recipe* is a symbolic template for an edit graph. Each recipe is specified as:

- `bindings` (references to clip metadata)
- `when` (eligibility predicate, e.g. vocal harmony requires a vocal stem)
- `weight_rules` (multipliers that indicate how common certain recipes should be)
- `graph` (a stem-aware declarative three-stage edit graph of the form `SEPARATE` → `PROCESS` → `MIX`)

Each recipe encodes a single post-production intent, such as adding modulation texture, applying vocal-leveling compression, or imparting a "vintage" lower-fidelity character. Recipes' tool-call arguments and certain structural slots (e.g. target stem) are resolved at instantiation. To share structure across recipes, we factor commonly co-occurring sub-chains into reusable units we call *motifs*: named, typed sub-graphs that can be expanded inline, e.g. the `retro_tilt_eq` motif expands to `high_shelf()` → `lowpass()`, capturing the joint high-frequency roll-off characteristic of a vintage sound. Each operator is also associated with a *sampling prior*, i.e. a distribution over its argument space calibrated so that draws produce audibly plausible edits rather than degenerate ones (e.g. a high-pass cutoff used for low-end cleanup is sampled in the 60–250Hz range, not from a uniform 20Hz–20kHz prior). Priors are defined per tool and per role: the same tool may carry a different prior when it appears in a different motif. We provide a web-based UI that allows authoring and auditioning all aspects.

Constraints RIME enforces two classes of constraints over candidate edit graphs. *Chain-order constraints* follow conventions about the order in which effect families are applied. We adopt the default ordering `EQ` → `Dynamics` → `Distortion` → `Modulation` → `Time-Based Effects` → `Leveling` and reject any candidate graph that violates it. Certain recipes carry recipe-specific orderings that override the default (e.g. vocal-harmony recipes place pitch processing upstream of dynamics) and here the recipe-level ordering is checked instead. Not every valid recipe is equally common in practice (e.g. modulation is rare on drums, spatial effects are more common in pop than classical, etc.). *Pattern-policy constraints* encode these priors as multiplicative weights on candidate recipes, conditioned on metadata of the clip (e.g. genre) and the target (e.g. instrument family).

Data Poisoning Real recordings rarely arrive in pristine condition, but often carry artifacts that post-production is tasked with removing. To capture this side of the workflow, we define two parallel families of recipes: *degradation recipes*, which introduce realistic recording artifacts (vocal sibilance, 50/60Hz mains hum, low-end rumble, buried or overpowering target stems), and matched *remediation recipes*, which apply the corresponding cleanup.

Generating and Subsampling Edit Plans The previous steps produce one edit graph per (clip, recipe) cell. To build a corpus, we execute it densely across the recipe catalog and the source dataset, and then subsample the resulting pool to a target size with controlled coverage. For each source clip, we instantiate every recipe whose `when` predicate is satisfied and sample multiple parameter draws per (clip, recipe) cell, generating a *permissible plan pool* which is large but highly redundant: some high-density (clip, recipe) cells contribute many near-duplicate plans that differ only in resampled argument values. We rank, stratify, and sample across the editorial categories that downstream evaluation cares about (harmony, delay, reverb, compression, modulation, etc.). We then draw the highest-ranked unselected candidates iteratively subject to a cardinality constraint: at most k plans per (clip, recipe) pair (we use $k = 1$).

Prompt Generation In music post-production, there are varying degrees of granularity with which you can describe the same set of edits. A sound engineer may be able to describe an edit with extreme specificity (e.g. "apply a highpass filter at 100 Hz"), while a musician may only be able to describe the edit in more general terms (e.g. "get rid of the low end rumble"). To simulate this, we iteratively generate prompts (see Algorithm 1). We set π = Gemini Flash Lite (Team et al., 2023) and set $K = 2$, yielding three abstraction levels per edit graph.

Algorithm 1 Prompt abstraction.

Require: edit graph G; language model π; abstraction depth $K \in \mathbb{N}$
Require: prompt templates $T_{\text{graph}}, T_{\text{abs}}$
$p_0 \sim \pi\big(\cdot \mid T_{\text{graph}}(G)\big)$ ▷ $G \to$ initial instruction
for $k = 1, \ldots, K$ **do**
$p_k \sim \pi\big(\cdot \mid T_{\text{abs}}(p_{k-1})\big)$ ▷ rewrite at increased abstraction
end for
return $(p_0, p_1, \ldots, p_K)$

3. Experiments

Generating Synthetic Data with RIME We use RIME to synthesize a triplet (input audio, edited audio, instruction) dataset starting from a corpus of full mixes with no available stem-level annotations. We draw a 10,000-clip random subsample from MTG-Jamendo (Bogdanov et al., 2019), in which instrument tags are largely absent, and tag every clip with MusicFlamingo (Ghosh et al., 2026), a multimodal music-understanding model. We retain only clips that contain both vocals and drums (these are most targeted by recipes), yielding 297 source clips. For these 297 clips, we run the plan-generation procedure and obtain a permissible-plan pool of 683,092 candidates and subsample this pool to ≈1,000 plans. For each selected plan we generate $K = 2$ rounds of abstraction-increasing prompts, yielding $1,000 \times 3 = 3,000$ unique triples. We additionally synthesize an artifact-removal split of 300 unique triples. For these examples, the agent receives the poisoned audio $\tilde{x}$ as input and is evaluated against the cleaned audio $\hat{x}$ as ground truth (rather than the original clip x).

Zero-Shot Agents using POEMS We design an agent to perform multi-turn POEMS tool calls using pretrained multimodal language models as follows:

1. Preliminary listening step: Listen to audio, determine whether the audio is a single source or a mixture.
2. Reasoning step: Obtain an advisory plan for what tools need to be called in what order
3. Tool calling step: Listen to the audio and look at the edits that have already been performed and determine what the next tool call should be.
4. Failure/recovery step: If the agent fails to make a properly-formed tool call, it can repeat the tool-calling attempt. For our experiments, we allow one retry.
5. Listen and repeat: the agent listens to the current audio and decides whether to complete or continue editing

We limit the agent to a maximum of 20 tool calling steps. We set up zero-shot agents using three multimodal models as orchestrators: GEMINI 3 FLASH (Team et al., 2023), GPT-4O MINI (Hurst et al., 2024), and GEMMA 3N (Team et al., 2024). We run the multi-step tool calling framework on the 3,000 baseline examples and 300 poisoning examples.

Evaluation Metrics We use two primary modes of evaluation: audio similarity, and edit-graph similarity. We use Frechet Audio Distance (Roblek et al., 2019) and Kernel Audio Distance (Chung et al., 2025) in the MERT (Yizhi et al., 2023) embedding space to measure the similarity between the direction of change of the agent's output vs. input compared to the direction of change of the ground-truth edited audio from RIME. For graphs, our primary metric is Graph F1, which is the arithmetic mean of four F1 scores, on graph *kind*, *operator*, *stem*, and *edge*.

4. Results

Post-Production Agents Struggle in Zero-Shot Settings No zero-shot agent we evaluate produces edits that are consistently aligned with the ground-truth target. Sometimes, they fail altogether: of 2,922 valid ground truth edits, GEMINI 3 FLASH successfully produces some output for 2,918, GPT-4O MINI for 2730, and GEMMA 3N for 2787. The failures are largely repeated tool call issues or looping up to the max steps without completion. Table 1 reports FAD, KAD, edit similarity (Δ Audio Sim, the cosine between δ_{gt} and δ_{out}, and Graph F1 averaged over the full benchmark. On audio, GPT-4O MINI reaches 0.611, GEMINI 3 FLASH 0.512, and GEMMA 3N 0.480. There is a notable gap between graph structure recovery and audio quality, even at rank level: GEMMA 3N strongly beats GEMINI 3 FLASH on graph F1 despite not doing so on audio. The FAD and KAD metrics both place GEMMA 3N in the lowest position , which is reasonable as it is a much smaller, open-weight baseline, but disagree with Δ Audio Sim on GEMINI 3 FLASH vs. GPT-4O MINI. We attribute the deviation between agent output and ground truth to struggles with effect parameterization, edits of increasing complexity, and specific edit categories.

Effect of Abstraction Layer The dominant axis of failure is how the prompt describes the edit. As prompts move from precise (AL0, with explicit parameter values) to evocative (AL2, with no numerical detail), nearly every metric we

track degrades. However, the magnitude and shape of that degradation is sharply model-specific. GPT-4o MINI falls from 0.733 at AL0 to 0.449 at AL2 (a 39% relative drop). GEMMA 3N falls further: 0.673 → 0.325 (52%). GEMINI 3 FLASH alone is approximately flat at 0.526 → 0.577 → 0.431 across the three levels, but starts already lower at AL0 than either competitor and gains essentially nothing from the explicit-parameter prompt. A consequence of these shapes is that the headline ranking changes at AL0. GEMMA 3N, which is the worst zero-shot model overall, is the second-best at AL0 (0.673, only 0.06 below GPT-4o MINI) and easily ahead of GEMINI 3 FLASH (0.526). Despite this, Graph F1 holds up reasonably well across abstraction levels, which we attribute to models selecting the right operators more often than they can parameterize them correctly.

Table 1. Primary evaluation metrics between agent output and ground truth. Slight negative KAD values indicate upstream numerical instability. AL = abstraction level

			Metrics				
	Category	N	FAD ↓	FAD_{inf} ↓	KAD ↓	Δ Aud. ↑	Gr. ↑
GPT-4o MINI	*Overall*	2730	0.046	0.042	0.085	0.611	0.845
	AL0	972	0.032	0.025	-0.020	0.733	0.929
	AL1	934	0.049	0.045	-0.004	0.628	0.849
	AL2	824	0.134	0.159	0.397	0.449	0.743
	drums	1006	0.126	0.100	0.265	0.564	0.850
	guitar	791	0.045	0.042	-0.060	0.688	0.855
	vocals	928	0.076	0.070	0.156	0.595	0.831
GEMINI 3 FLASH	*Overall*	2918	0.120	0.120	0.228	0.511	0.702
	AL0	970	0.147	0.141	0.278	0.526	0.701
	AL1	974	0.131	0.116	0.120	0.577	0.761
	AL2	974	0.136	0.181	0.240	0.431	0.634
	drums	1025	0.240	0.254	0.428	0.507	0.751
	guitar	843	0.136	0.125	0.175	0.479	0.602
	vocals	1044	0.168	0.188	0.242	0.540	0.736
GEMMA 3N	*Overall*	2787	0.227	0.230	0.573	0.480	0.803
	AL0	966	0.037	0.037	-0.026	0.673	0.917
	AL1	902	0.213	0.213	0.502	0.432	0.778
	AL2	919	0.608	0.635	2.098	0.325	0.707
	drums	999	0.210	0.183	0.691	0.499	0.827
	guitar	833	0.252	0.258	0.318	0.508	0.829
	vocals	949	0.452	0.453	0.992	0.436	0.755

Edits are Non-Uniformly Challenging Table 1 shows that every model has a target stem it cannot edit well, and the three models do not necessarily agree on which one. GPT-4o MINI and GEMINI 3 FLASH are weakest on drums. GEMMA 3N, by contrast, is weakest on vocals by a substantial margin. A second pattern emerges when we stratify by graph complexity. Figure 1 also shows that models generally struggle as edit graphs contain more operations. Specifically, in abstraction layers 1 and 2 (where parameterizations or even effects, sometimes are not explicitly defined in the prompt), there is a clear downward trend. Interestingly, the graph-level metric degrades less, suggesting that even long chains can be inferred; however the baseline depends on the abstraction level. Together with the abstraction analysis, this suggests that zero-shot agents fail at parameter inference from natural language descriptions, struggle with operator inference at higher abstraction levels, and such parameter inference errors may compound multiplicatively along the edit chain, resulting in significant deviations in the audio output.

Table 2. Metrics for poisoning experiments (Gr. = Graph F1).

Model	AL0		AL1		AL2	
	Δ Aud. ↑	Gr. ↑	Δ Aud. ↑	Gr. ↑	Δ Aud. ↑	Gr. ↑
GEMINI 3 FLASH	0.892	0.823	0.702	0.753	0.475	0.598
GPT-4o MINI	0.987	0.879	0.658	0.775	0.476	0.656
GEMMA 3N	0.885	0.826	0.415	0.594	0.313	0.494

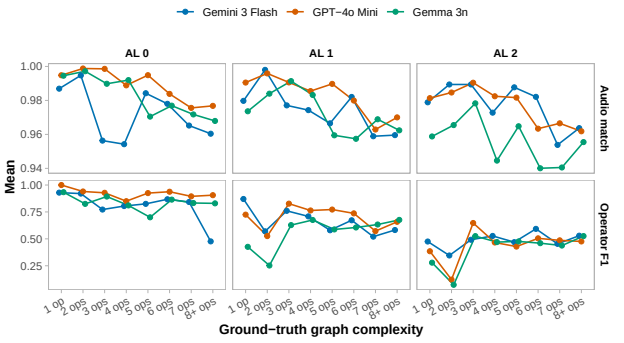

Figure 1. Audio similarity and operator F1 for edit graphs based on number of operations and abstraction level.

Poisoning The artifact-removal split tests a different ability: identifying *what is wrong* with a track and reaching for the correct corrective tools, rather than executing on a request. Results are shown in Table 2. At AL0, where the corrective tool is often named, all three models reach audio ≥ 0.85 and operator F1 ≥ 0.8. At AL2, audio match scores drop to 0.3–0.5 and operator F1 falls below 0.6 for 2/3 models. This suggests that even structurally simple chains (most poisoning recipes are 1–3 operations) are not reliably reconstructed under ambiguity.

5. Conclusion

The dominant paradigm in music ML is one-shot generation. The dominant paradigm in studio practice is iteration. Reconciling these competing visions requires data and infrastructure, and our work presents a scalable first step on this front. A first step always has limitations: our dependency on human-coded recipes, a limited operator pool, no external baselines yet available, and no perfect metrics for this problem. The principal limitation of our work is that we can only test zero-shot agents, since a post-training recipe has not yet been established for this task. We invite the community to build on our work to do so.

References

Anthropic, P. Introducing the model context protocol. *https://www. anthropic. com/news/model-contextprotocol*, 2024.

Bogdanov, D., Won, M., Tovstogan, P., Porter, A., and Serra, X. The mtg-jamendo dataset for automatic music tagging. In *Machine learning for music discovery workshop, international conference on machine learning (ICML 2019)*, pp. 1–3. Long Beach, CA, United States, 2019.

Chu, A., O'Reilly, P., Barnett, J., and Pardo, B. Text2fx: Harnessing clap embeddings for text-guided audio effects. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2025.

Chung, Y., Eu, P., Lee, J., Choi, K., Nam, J., and Chon, B. S. Kad: No more fad! an effective and efficient evaluation metric for audio generation. *arXiv preprint arXiv:2502.15602*, 2025.

Doh, S., Koo, J., Martínez-Ramírez, M. A., Choi, W., Liao, W.-H., Wu, Q., Nam, J., and Mitsufuji, Y. Llm2fx-tools: Tool calling for music post-production. In *The Fourteenth International Conference on Learning Representations*, 2026.

Ellis, D. P., Fonseca, E., Weiss, R. J., Wilson, K., Wisdom, S., Erdogan, H., Hershey, J. R., Jansen, A., Moore, R. C., and Plakal, M. Recomposer: Event-roll-guided generative audio editing. *arXiv preprint arXiv:2509.05256*, 2025.

Ghosh, S., Goel, A., Koroshinadze, L., Lee, S.-g., Kong, Z., Santos, J. F., Duraiswami, R., Manocha, D., Ping, W., Shoeybi, M., et al. Music flamingo: Scaling music understanding in audio language models. In *The Fourteenth International Conference on Learning Representations*, 2026.

Han, B., Dai, J., Hao, W., He, X., Guo, D., Chen, J., Wang, Y., Qian, Y., and Song, X. Instructme: an instruction guided music edit framework with latent diffusion models. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pp. 5835–5843, 2024.

Hurst, A., Lerer, A., Goucher, A. P., Perelman, A., Ramesh, A., Clark, A., Ostrow, A., Welihinda, A., Hayes, A., Radford, A., et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Imort, J., Fabbro, G., Martinez Ramirez, M. A., Uhlich, S., Koyama, Y., and Mitsufuji, Y. Distortion audio effects: Learning how to recover the clean signal. In *Ismir 2022 Hybrid Conference*, 2022.

Jia, Y., Chen, Y., Zhao, J., Zhao, S., Zeng, W., Chen, Y., and Qin, Y. Audioeditor: A training-free diffusion-based audio editing framework. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2025.

Lan, Z., Hao, Y., and Zhao, M. Declarative audio editing with audio language model. In *The Fourteenth International Conference on Learning Representations*, 2026.

Liang, J., Zhang, H., Liu, H., Cao, Y., Kong, Q., Liu, X., Wang, W., Plumbley, M. D., Phan, H., and Benetos, E. Wavcraft: Audio editing and generation with large language models. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*.

Liang, J., Chen, Y., Yuan, Y., Jia, D., Zhuang, X., Chen, Z., Wang, Y., and Wang, Y. Audiomorphix: Training-free audio editing with diffusion probabilistic models. *arXiv preprint arXiv:2505.16076*, 2025.

Manor, H. and Michaeli, T. Zero-shot unsupervised and text-based audio editing using ddpm inversion. In *International Conference on Machine Learning*, pp. 34603–34629. PMLR, 2024.

Rice, M., Steinmetz, C. J., Fazekas, G., and Reiss, J. D. General purpose audio effect removal. In *2023 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pp. 1–5. IEEE, 2023.

Roblek, D., Kilgour, K., Sharifi, M., and Zuluaga, M. Fr\'echet audio distance: A reference-free metric for evaluating music enhancement algorithms. In *Proc. Interspeech*, pp. 2350–2354, 2019.

Rouard, S., Massa, F., and Défossez, A. Hybrid transformers for music source separation. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2023.

Sobot, P. Pedalboard, July 2021. URL https://doi.org/10.5281/zenodo.7817838.

Team, G., Anil, R., Borgeaud, S., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A. M., Hauth, A., Millican, K., et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Team, G., Mesnard, T., Hardin, C., Dadashi, R., Bhupatiraju, S., Pathak, S., Sifre, L., Rivière, M., Kale, M. S., Love, J., et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

Tsai, F.-D., Wu, S.-L., Kim, H., Chen, B.-Y., Cheng, H.-C., and Yang, Y.-H. Audio prompt adapter: Unleashing music editing abilities for text-to-music with lightweight finetuning. In *Ismir 2024 Conference*, 2024.

Wang, Y., Ju, Z., Tan, X., He, L., Wu, Z., Bian, J., et al. Audit: Audio editing by following instructions with latent diffusion models. *Advances in Neural Information Processing Systems*, 36:71340–71357, 2023.

Yang, Y., Li, H., Li, T., Cao, B., Zhang, X., Chen, L., and Liu, Q. Melodia: Training-free music editing guided by attention probing in diffusion models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 2209–2217, 2026.

Yizhi, L., Yuan, R., Zhang, G., Ma, Y., Chen, X., Yin, H., Xiao, C., Lin, C., Ragni, A., Benetos, E., et al. Mert: Acoustic music understanding model with large-scale self-supervised training. In *The Twelfth International Conference on Learning Representations*, 2023.

Zhang, Y., Ikemiya, Y., Xia, G., Murata, N., Martínez-Ramírez, M. A., Liao, W.-H., Mitsufuji, Y., and Dixon, S. Musicmagus: zero-shot text-to-music editing via diffusion models. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pp. 7805–7813, 2024.