MixingBuddy: A Multimodal LLM for Mix Critique and Advice

Pratham Vadhulas, Alexander Lerch

***Abstract*—this is the abstract. here is some more text. here is some more text.**

I. Introduction

Automatic Mixing, a key subfield of Music Informatics Research (MIR), aims to automate the complex and subjective task of music mixing. This area of study is pivotal to the modern music production and audio engineering market. To date, research in this field has made significant progress, largely by leveraging deep learning. Sophisticated models, such as U-Nets or generative frameworks, have been developed to make mixing systems accurate (predicting parameters that match professional mixes), controllable (allowing for high-level parameters to be set), and diverse (accomadating different genres and styles).

However, a critical limitation of these approaches is their "black box" nature. They can perform the mix, but they cannot explain their reasoning. The recent promise of large language models (LLMs) and multi-modal "agentic" systems introduces a new, necessary paradigm: explainability. We can now envision a tool that reasons about and discusses a mix.

This potential for co-creative, linguistic feedback brings us to our core research question: To what extent can an audio language model, when given a flawed mix, provide correct and useful advice?

II. Related Work

Early automatic mixing research centered on systems that captured expert knowledge through explicit mixing rules and heuristics. [1] developed an autonomous mixing system based on knowledge engineering principles. [2] employed probabilistic expert systems, a formal knowledge-based approach from early AI research, for automatic music production. Subsequent work explored machine learning techniques for instrument-specific effects, such as [3]'s approach to intelligent artificial reverberation application. [4] presented intelligent multitrack reverberation based on hinge-loss Markov random fields, demonstrating the application of statistical modeling tools to mixing tasks. [5] described a statistical approach to automated offline dynamic processing in the audio mastering process, further exemplifying the use of traditional machine learning methods. [6] introduced a framework that uses genetic optimization with timbral similarity measures. While these methods provided interpretable control and domain-specific optimization, they were limited in their ability to generalize across diverse musical styles and lacked the flexibility to adapt to unseen mixing scenarios.

The advent of deep learning brought significant advances in automatic mixing, with models capable of learning complex mappings from raw audio to mixing parameters. [7] introduced Wave-U-Net autoencoders for automatic mixing, demonstrating that end-to-end neural architectures could produce professional-quality mixes. [8] further advanced the field with a differentiable mixing console incorporating neural audio effects, enabling gradient-based optimization of mixing parameters. These deep learning approaches achieved notable success in terms of accuracy (matching professional mixes), controllability (allowing high-level parameter adjustment), and diversity (accommodating different genres and styles). However, a critical limitation of these systems is their "black box" nature: while they can perform the mix, they cannot explain their reasoning or provide linguistic feedback about mixing decisions.

Recognizing the need to bridge the semantic gap between audio processing and human understanding, researchers began exploring word-embedding approaches that link natural language descriptions to audio effect parameters. [9] demonstrated word embeddings for automatic equalization in audio mixing, while [10] developed Text2FX, which harnesses CLAP embeddings for text-guided audio effects. Early semantic mixing approaches [11] laid the groundwork for understanding how high-level, semantic knowledge could inform mixing decisions. These methods represented initial attempts to make mixing systems more interpretable and user-friendly by connecting linguistic descriptions to audio processing parameters.

Building on language-audio integration, recent work has explored prompt-driven interfaces that map natural language instructions directly to mixing tasks. [12] investigated whether large language models can predict audio effects parameters from natural language, while [13] developed SonicMaster, a controllable all-in-one music restoration and mastering system. [14] introduced MixAssist, an audio-language dataset for co-creative AI assistance in music mixing, demonstrating the potential for collaborative human-AI mixing workflows. These approaches represent an evolution toward more natural interaction paradigms, where users can express mixing intentions in natural language rather than manipulating low-level parameters.

Multimodal audio-language models have emerged as a powerful paradigm for combining audio understanding with language reasoning capabilities. One architectural approach, direct tokenization (also known as the unified approach), con-

Pratham Vadhulas is with the Georgia Institute of Technology, Atlanta, GA, USA (email: pvadhulas3@gatech.edu).

Alexander Lerch is with the Georgia Institute of Technology, Atlanta, GA, USA (email: alexander.lerch@gatech.edu).

verts raw audio into discrete tokens via audio codecs and extends the LLM vocabulary to include these audio tokens. This enables the language model to process audio and text within a unified framework. Key works in this direction include [15]'s AudioPaLM, [16]'s LauraGPT, and [17]'s SpeechGPT. The unified approach offers the advantage of treating audio and text as first-class citizens within the same model architecture, potentially enabling more seamless cross-modal reasoning.

An alternative architectural paradigm, the cascade (or feature extraction) approach, uses audio-specific encoders and decoders with the LLM serving as a central backbone. In this framework, audio is first encoded into feature representations that are then processed by the language model, which can generate text responses or guide audio generation. Examples include [18]'s M^{2}UGen and [19]'s Listen, Think, and Understand (LTU). The cascade approach allows for specialized audio processing while leveraging the reasoning capabilities of large language models, making it particularly relevant for tasks requiring both audio understanding and linguistic explanation, such as mix critique and advice.

Multimodal audio-language models show promise for explainable audio processing, but their application to mix critique and advice remains largely unexplored. Our work addresses this gap by leveraging multimodal audio-language models to provide linguistic feedback and reasoning about mixes. This positions our work as a step toward explainable, co-creative mixing systems that bridge the gap between automated processing and human understanding.

III. Methodology

To address this gap, we propose a multimodal audio-language model that takes a flawed mix as input and generates structured textual feedback identifying mixing flaws and suggesting corrective gain adjustments. Our approach focuses on relative gain relationships among multitrack stems. This focus allows us to investigate whether the model can learn the relative nature of gain relationships. While also providing a tractable starting point before extending to more complex mixing parameters such as equalization or dynamic processing. the methodology section is organized as follows III-A Dataset, III-B Architecture, III-C Training Strategy.

A. Dataset

Our methodology leverages the MUSDB18HQ dataset [20], a high-quality multitrack audio dataset, as the foundation for our training data. We augment this dataset to generate flawed mixes for both Supervised Fine-Tuning (SFT) and Direct Preference Optimization (DPO) training phases. The dataset provides four stems for each track: 'bass', 'drums', 'other', and 'vocals'.

A key consideration in gain balancing is its relative nature. For instance, a bass stem is too loud in comparison to the other stems, not by some absolute value, but by a relative value. To address this ambiguity and guide the model towards a specific solution, we introduce the concept of an 'anchor' stem. The anchor stem is a reference track whose gain is assumed to be correct. By providing an anchor stem, we constrain the problem, allowing the model to infer the intended gain adjustment.

The 'other' stem often contains a wide variety of instruments and sounds, leading to inconsistency across different tracks. Therefore, for the purposes of controlled experiments, we exclude the 'other' stem from being the target of gain alterations or being used as an anchor.

1) SFT Dataset Synthesis: Our SFT training data is synthetically generated by creating diverse instruction-response pairs through a two-stage synthesis process: flawed mix generation and response templating. Figure 1 illustrates the complete synthesis pipeline.

Flawed Mix Generation. We begin by segmenting each track into 10-second chunks to ensure manageable audio lengths for processing. For each chunk, we randomly select a flaw category from five possible categories: *no error*, *quiet*, *very quiet*, *loud*, and *very loud*, each with equal probability (20%). We then randomly select a target stem from the available stems (bass, drums, or vocals). Based on the selected flaw category, we inject a gain error into the target stem by applying a gain adjustment within predefined decibel ranges: very quiet (-12 to -6 dB), quiet (-6 to -3 dB), loud ($+3$ to $+6$ dB), and very loud ($+6$ to $+12$ dB). For the *no error* category, we apply a small random gain adjustment between -3 and $+3$ dB to introduce natural variation while maintaining the overall balanced nature of the mix. The flawed mix is then created by summing all stems, including the modified target stem.

Response Synthesis. To generate corresponding textual responses, we employ a category-driven templating approach. Figure 2 illustrates this process. For each flaw category, we maintain a pool of 10 semantically equivalent response templates that vary in wording. A response template is randomly selected from the category-specific template pool based on the injected flaw category. Each template contains placeholders for dynamic variables such as the target stem name and suggested gain adjustment values (specified as {`min_gain_db`} and {`max_gain_db`}). We use a range rather than a value because we assume the large language models are not accurate numerical value predictors, they are simply token predictors. Enhancing the model's ability to predict numerical values is a future direction. These placeholders are automatically populated using the ground truth information: the actual target stem that received the error and the specific gain adjustment value (in dB) that was applied. For example, a *quiet* category template might state: "The {target_stem} is a little too quiet. Increase the {target_stem} level between {min_gain_db} and {max_gain_db} dB to balance the mix." This process ensures that each response reflects the mixing flaw present in the corresponding flawed mix while maintaining linguistic diversity through 10 template variations per flaw category.

Instruction Generation. Each training sample includes an instruction that provides context to the model. Instructions are generated by randomly selecting from a pool of 10 semantically equivalent instruction templates that vary in wording while maintaining the same semantic content. Each template contains placeholders for chunk-specific information: the segment duration ({`duration_sec`}), the list of

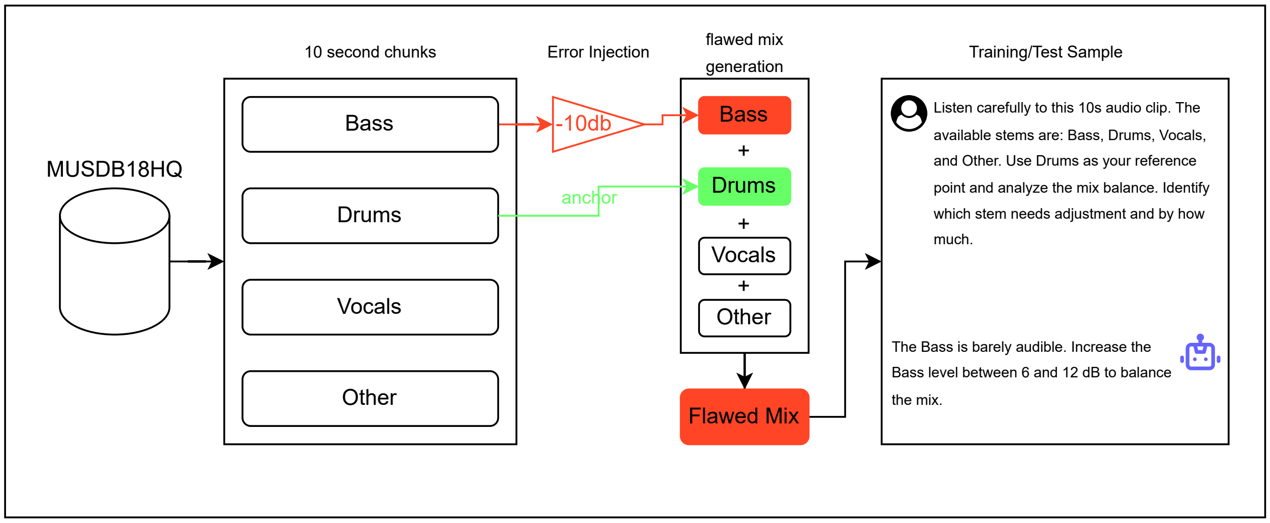
10 second chunks
Error Injection
flawed mix generation
Training/Test Sample
MUSDB18HQ
Bass
Drums
Vocals
Other
-10db
anchor
Bass
+
Drums
+
Vocals
+
Other
Flawed Mix
Listen carefully to this 10s audio clip. The available stems are: Bass, Drums, Vocals, and Other. Use Drums as your reference point and analyze the mix balance. Identify which stem needs adjustment and by how much.
The Bass is barely audible. Increase the Bass level between 6 and 12 dB to balance the mix.

Fig. 1. Overview of the training and test sample synthesis pipeline. The process involves random flaw selection, ground truth label generation from MUSDB18, response template selection, and placeholder population to create synthesized responses.

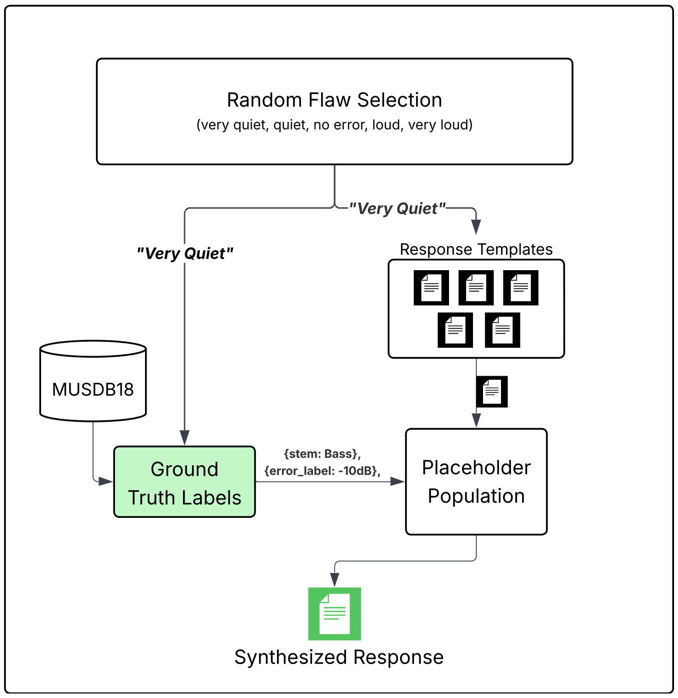

Fig. 2. Response synthesis process. A flaw category is randomly selected, which determines the response template pool. The selected template is then populated with ground truth information including the target stem name and gain adjustment values.

available stems ({`stems_present`}), and the anchor stem designation ({`anchor_stem`}). For example, an instruction template might state: "Listen carefully to this {duration_sec}s audio clip. The available stems are: {stems_present}. Use {anchor_stem} as your reference point and analyze the mix balance. Identify which stem needs adjustment and by how much." This approach ensures linguistic diversity in instructions while maintaining consistent semantic meaning across all training samples.

This synthesis pipeline produces a large-scale dataset of approximately 100,000 (instruction, response) triplets, including both training and test sets. The dataset diversity stems from multiple augmentation dimensions: 3 target stems (bass, drums, vocals), 2 possible anchor stems per target (selected from the remaining stems), and 5 error categories. Figure 3 illustrates the augmentation strategy and resulting dataset composition. Each triplet contains a known gain imbalance (or balanced state for *no error* samples) and a corresponding response that provides accurate feedback and corrective suggestions.

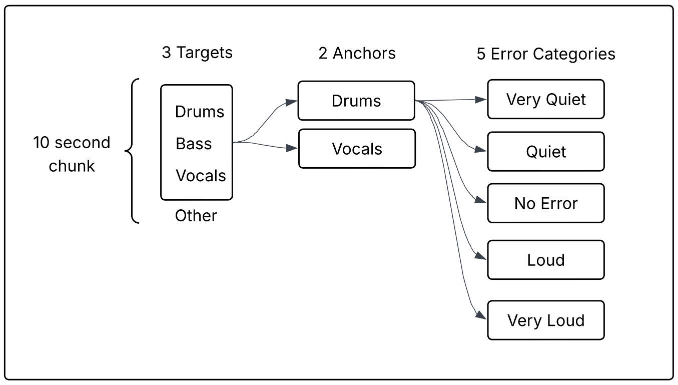

Fig. 3. Dataset augmentation strategy showing the combination of target stems, anchor stems, and error categories that contribute to the final dataset of approximately 100,000 samples (including training and test sets).

B. Architecture

Our architecture implements a multimodal framework that processes 10-second audio segments paired with textual instructions to generate accurate feedback addressing gain balancing anomalies. Following the audio prefixing approach demonstrated in multimodal language models [19], we concatenate audio embeddings with text embeddings before passing them to the large language model. This design requires training an audio projection layer to align the encoder output embeddings (1024 dimensions) with the LLM input embedding space (typically 3584 dimensions for Qwen2-7B-Instruct). While alternative modality alignment techniques exist [18], audio prefixing provides an effective starting point for our application. The architecture employs MERT-v1-330M as the audio encoder, a trainable MLP projection layer, and Qwen2-7B-Instruct as the backbone language model. Figure 4 illustrates the complete architecture.

1) Audio Encoder: We employ the MERT-v1-330M encoder [] as our audio feature extractor. The encoder produces 25 transformer layers of representations, from which we extract a weighted average using learnable layer weights. This weighted aggregation allows the model to adaptively emphasize different hierarchical levels of audio representation during training. The encoder outputs 1024-dimensional embeddings at each time step.

2) Audio Projection: To bridge the gap between audio encoder outputs and LLM input embeddings, we employ a multilayer perceptron (MLP) projection network that maps 1024-dimensional audio embeddings to embeddings compatible with the LLM's input space. Pilot experiments demonstrated that increasing the MLP depth or incorporating transformer layers or other complex architectures did not yield performance improvements. Consequently, we adopt a simple MLP architecture with four hidden layers, each containing 2048 units, using GELU activation functions, layer normalization, and a dropout rate that begins at 0.3 and is reduced to 0.1 once training plateaus. We do not employ residual connections in the projection network. During training, we incorporate an auxiliary loss with a weight of 0.05 to facilitate learning of the projection mapping. The projected audio embeddings are concatenated with text token embeddings and passed to the language model for processing.

C. Training Strategy

We employ supervised fine-tuning (SFT) to train our models on the synthesized instruction-response pairs. To assess the contribution of different architectural components, we evaluate two primary model configurations that differ in the extent of parameter adaptation.

1) Model Configurations:

a) Audio-Only Projection (Baseline): The baseline configuration evaluates whether a learnable projection layer alone is sufficient to map audio features to the language model's semantic space. This configuration employs a frozen Qwen2-7B-Instruct backbone with 4-bit quantization, a frozen MERT-v1-330M audio encoder, and a trainable MLP projection network. The projection network consists of four hidden layers with dimensions [2048, 4096, 4096, 2048], using GELU activation functions, layer normalization, and dropout regularization. This constrained setup forces the model to learn the audio-to-text mapping entirely through the projection layer, providing insight into the minimum architectural requirements for the task.

b) Joint Adaptation (Projection + LoRA): The joint adaptation configuration extends the baseline by incorporating Low-Rank Adaptation (LoRA) adapters on all linear layers of the language model. Specifically, LoRA adapters are applied to the attention projection layers (`q_proj`, `k_proj`, `v_proj`, `o_proj`) and the feed-forward network layers (`gate_proj`, `up_proj`, `down_proj`). The LoRA adapters use a rank $r = 8$, scaling factor $\alpha = 16$, and dropout rate of 0.1. This configuration allows the language model to adapt its internal representations to better process the audio-conditioned inputs, potentially improving performance at the cost of increased parameter count and training complexity.

2) Implementation Details: All models are implemented using PyTorch [**?**] with the HuggingFace Transformers library [**?**] and the Parameter-Efficient Fine-Tuning (PEFT) framework [**?**]. To enable efficient training of large language models, we employ 4-bit NormalFloat (NF4) quantization [**?**] with double quantization and bfloat16 compute type through the bitsandbytes library [**?**]. The audio encoder remains frozen throughout training, while the projection layer and, when applicable, LoRA adapters are trained end-to-end.

3) Training Protocol: All models are trained using the AdamW optimizer [**?**] with a learning rate of 1×10^{-4}. We employ a cosine learning rate schedule with a warmup period covering 1% of the total training steps. Training is conducted with an effective batch size of 16, achieved through a per-device batch size of 2 and gradient accumulation over 8 steps. Mixed precision training is enabled using bfloat16 to reduce memory consumption and accelerate training.

The training objective combines the standard causal language modeling (CLM) loss with an auxiliary projection loss weighted at 0.05. This auxiliary loss ensures strong gradient flow to the projection layer, which is particularly important for the baseline configuration where the projection layer is the sole trainable component. Models are trained for up to 100 epochs with early stopping based on validation loss to prevent overfitting.

IV. Evaluation

We evaluate the two model configurations described in Section III-C to assess their performance on the mixing advice task. The evaluation methodology focuses on extracting structured information from the model's free-form text responses and comparing it against ground truth annotations.

A. Evaluation Methodology

We employ a pattern-matching based evaluation approach to assess the accuracy of generated mixing advice. This methodology extracts structured information from the model's free-form text responses and compares it against ground truth annotations.

1) Pattern Matching for Response Analysis: The evaluation process analyzes each generated response to identify two key components: (1) the target stem that requires adjustment, and (2) the direction of the required adjustment (increase, decrease, or no adjustment needed). To extract this information, we

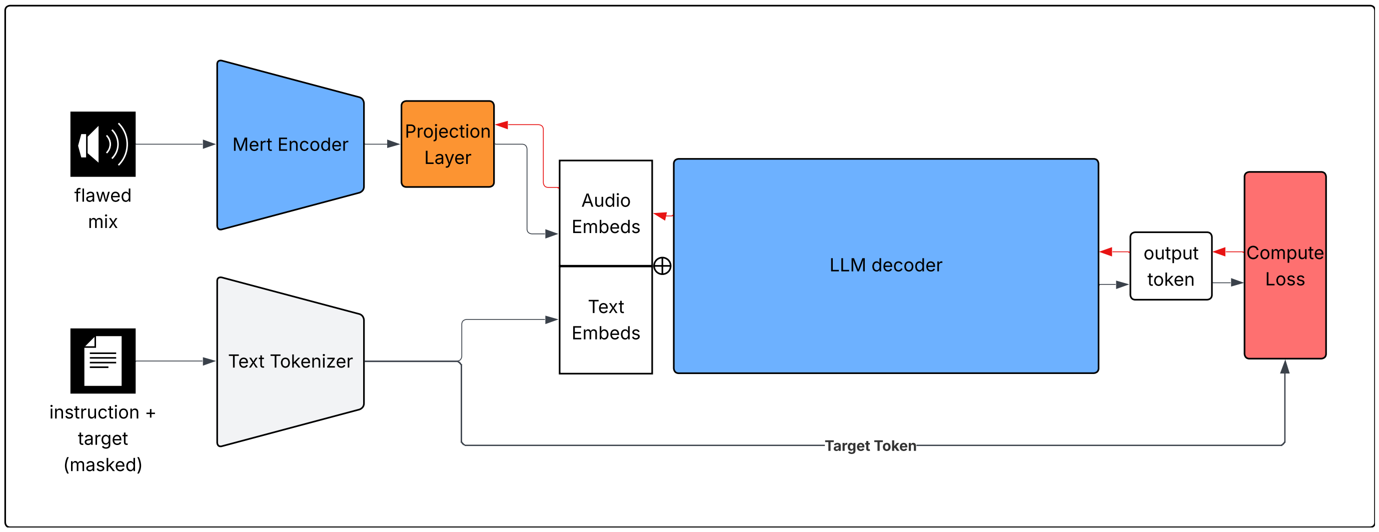

Fig. 4. Architecture overview showing the multimodal framework with audio encoder, projection layer, and language model components.

employ keyword-based pattern matching that searches for semantic indicators within the generated text.

The pattern matching system uses three sets of keywords derived from the response template variations used during training:

- **Increase keywords**: Indicators for stems that are too quiet and need to be boosted (e.g., “increase”, “boost”, “raise”, “too quiet”, “needs more volume”).
- **Decrease keywords**: Indicators for stems that are too loud and need to be reduced (e.g., “reduce”, “decrease”, “lower”, “too loud”, “overpowering”).
- **No error keywords**: Indicators that the mix is well-balanced and requires no adjustments (e.g., “well-balanced”, “no adjustments needed”, “properly balanced”).

The evaluation algorithm processes each generated response by:

1) Splitting the text into sentences for localized analysis.
2) Identifying sentences that contain both a stem name (vocals, drums, bass, or other) and a directional keyword.
3) Extracting the identified stem and direction from the matched sentence.
4) For cases where no problem stem is identified, checking for “no error” keywords to determine if the model correctly identified a balanced mix.
5) Detecting contradictions where the same stem is associated with both increase and decrease keywords, which results in an incorrect classification.

2) Accuracy Metrics: We evaluate model performance using three complementary accuracy metrics:

- **Stem accuracy**: The percentage of predictions where the identified target stem matches the ground truth stem that received the gain adjustment.
- **Direction accuracy**: The percentage of predictions where the identified adjustment direction (increase, decrease, or none) matches the expected direction based on the error category.
- **Both correct**: The percentage of predictions where both the stem and direction are correctly identified, representing the strictest accuracy measure.

For the “no error” category, where no stem adjustment is needed, stem accuracy is defined to match direction accuracy, as correctly identifying that no adjustment is needed satisfies both criteria.

3) Performance Breakdowns: To gain deeper insights into model behavior, we analyze performance across multiple dimensions:

- **By error category**: Performance breakdown across the five error categories (no error, quiet, very quiet, loud, very loud) to identify which types of mixing issues are most challenging.
- **By target stem**: Performance breakdown across the three target stems (vocals, drums, bass) to assess whether the model has biases toward certain instruments.
- **By direction type**: Performance breakdown by the required adjustment direction (increase, decrease, no error) to evaluate directional prediction capabilities.

For each breakdown dimension, we compute both micro-averaged accuracies (overall performance) and macro-averaged accuracies (average across all classes, giving equal weight to each class regardless of sample distribution).

4) Statistical Significance Analysis: To determine whether performance differences between model configurations are statistically significant, we employ appropriate statistical tests. For binary classification metrics (correct vs. incorrect), we use McNemar’s test [**?**] to compare paired predictions between configurations. For continuous accuracy metrics, we perform paired t-tests or Wilcoxon signed-rank tests depending on the distribution of the data. Statistical significance is assessed at the $p < 0.05$ level, with Bonferroni correction applied when conducting multiple comparisons to control for family-wise error rate.

TABLE I
PERFORMANCE BREAKDOWN BY ERROR CATEGORY FOR THE PROJECTION + LINEAR LAYERS CONFIGURATION.

Category	N	Both	Stem	Dir.
very loud	30	90.0%	90.0%	90.0%
quiet	16	56.2%	75.0%	62.5%
very quiet	18	50.0%	50.0%	77.8%
no error	20	30.0%	30.0%	30.0%
loud	16	25.0%	56.2%	25.0%
Macro Avg.	–	**50.25%**	**60.25%**	**57.06%**

TABLE II
PERFORMANCE BREAKDOWN BY TARGET STEM FOR THE PROJECTION + LINEAR LAYERS CONFIGURATION.

Stem	N	Both	Stem	Dir.
drums	23	65.2%	73.9%	65.2%
bass	40	62.5%	70.0%	62.5%
vocals	37	40.5%	48.6%	56.8%
Macro Avg.	–	**56.09%**	**64.19%**	**61.49%**

TABLE III
PERFORMANCE BREAKDOWN BY DIRECTION TYPE FOR THE PROJECTION + LINEAR LAYERS CONFIGURATION.

Direction	N	Both	Stem	Dir.
decrease	46	67.4%	78.3%	67.4%
increase	34	52.9%	61.8%	70.6%
no error	20	30.0%	30.0%	30.0%
Macro Avg.	–	**50.11%**	**56.67%**	**55.99%**

V. RESULTS

We present the experimental results for the model configurations described in Section III-C. The evaluation methodology is detailed in Section IV-A.

A. Projection + Linear Layers Configuration

We first evaluate the configuration employing a trainable projection layer with full fine-tuning of the language model's linear layers. This configuration represents a more expressive adaptation strategy compared to the projection-only baseline, allowing the model to adjust its internal representations to better process audio-conditioned inputs.

1) Overall Performance: On a test set of 100 samples, the projection + linear layers configuration achieves 55.00% accuracy for the joint task of correctly identifying both the target stem and adjustment direction. The model demonstrates 63.00% accuracy for stem identification and 61.00% accuracy for direction prediction. These results indicate that while the model can identify problematic stems with reasonable accuracy, the joint task of correctly identifying both components remains challenging.

2) Performance by Error Category: Table I presents the performance breakdown across the five error categories. The model shows strong performance on the *very loud* category, achieving 90.0% accuracy for both stem and direction identification, suggesting that extreme loudness issues are more readily detectable. Performance on the *quiet* category is moderate (56.2% both correct), while the *very quiet* category achieves 50.0% accuracy. The model struggles most with the *loud* category (25.0% both correct) and the *no error* category (30.0% both correct), indicating challenges in distinguishing moderate imbalances and correctly identifying well-balanced mixes.

3) Performance by Target Stem: Table II shows the performance breakdown across the three target stems. The model demonstrates relatively balanced performance across stems, with bass and drums achieving similar accuracies (62.5% and 65.2% both correct, respectively). Vocals show lower performance (40.5% both correct), suggesting that vocal stem identification may be more challenging, potentially due to the more complex spectral characteristics or contextual dependencies of vocal tracks.

4) Performance by Direction Type: Table III presents the performance breakdown by adjustment direction. The model shows stronger performance for *decrease* predictions (67.4% both correct) compared to *increase* predictions (52.9% both correct), suggesting that identifying when stems are too loud may be easier than identifying when they are too quiet. The *no error* category remains challenging (30.0% both correct), consistent with the error category analysis.

REFERENCES

[1] E. Pérez-González and J. D. Reiss, "A knowledge-engineered autonomous mixing system," in *Audio Engineering Society Convention 135*, Oct. 2013.

[2] G. Bocko, M. F. Bocko, D. Headlam, J. Lundberg, and G. Ren, "Automatic music production system employing probabilistic expert systems," *journal of the audio engineering society*, 2010.

[3] E. Chourdakis and J. Reiss, "A machine-learning approach to application of intelligent artificial reverberation," *Journal of the Audio Engineering Society*, vol. 65, p. 56–65, Feb. 2017.

[4] A. L. Benito and J. D. Reiss, "Intelligent Multitrack Reverberation Based on Hinge-Loss Markov Random Fields," in *Proceedings of the 2017 AES International Conference on Semantic Audio*, June 2017.

[5] M. Hilsamer and S. Herzog, "A statistical approach to automated offline dynamic processing in the audio mastering process," in *Proc. of the 17th Int. Conference on Digital Audio Effects (DAFx-14)*, September 2014.

[6] B. Kolasinski, "A framework for automatic mixing using timbral similarity measures and genetic optimization," vol. 3, 05 2008.

[7] E. Chourdakis and J. D. Reiss, "Automatic music signal mixing system based on one-dimensional wave-u-net autoencoders," *EURASIP Journal on Audio, Speech, and Music Processing*, 2022.

[8] C. J. Steinmetz, J. Pons, S. Pascual, and J. Serrà, "Automatic multitrack mixing with a differentiable mixing console of neural audio effects," Oct. 2020.

[9] S. Venkatesh, D. Moffat, and E. R. Miranda, "Word embeddings for automatic equalization in audio mixing," *Journal of the Audio Engineering Society*, vol. 70, p. 753–763, Nov. 2022.

[10] A. Chu, P. O'Reilly, J. Barnett, and B. Pardo, "Text2fx: Harnessing clap embeddings for text-guided audio effects," Feb. 2025.

[11] E. Chourdakis and J. Reiss, "A semantic approach to autonomous mixing," 2016.

[12] S. Doh, J. Koo, M. A. Martínez-Ramírez, W.-H. Liao, J. Nam, and Y. Mitsufuji, "Can large language models predict audio effects parameters from natural language?" Jul. 2025.

[13] J. Melechovsky, A. Mehrish, and D. Herremans, "Sonicmaster: Towards controllable all-in-one music restoration and mastering," Aug. 2025.

[14] M. P. Clemens and A. Marasovic, "Mixassist: An audio-language dataset for co-creative AI assistance in music mixing," 2025.

[15] P. K. Rubenstein, C. Asawaroengchai, D. D. Nguyen, and et al., "Audiopalm: A large language model that can speak and listen," Jun. 2023.

[16] Z. Du, J. Wang, Q. Chen, Y. Chu, Z. Gao, and et al., "Lauragpt: Listen, attend, understand, and regenerate audio with gpt," Jul. 2024.

[17] D. Zhang, S. Li, X. Zhang, J. Zhan, P. Wang, Y. Zhou, and X. Qiu, "Speechgpt: Empowering large language models with intrinsic cross-modal conversational abilities," May 2023.

[18] S. Liu, A. S. Hussain, Q. Wu, C. Sun, and Y. Shan, "M^{2}ugen: Multi-modal music understanding and generation with the power of large language models," Dec. 2024.

[19] Y. Gong, H. Luo, A. H. Liu, L. Karlinsky, and J. Glass, "Listen, think, and understand," Feb. 2024.

[20] Z. Rafii, A. Liutkus, F.-R. St"oter, S. I. Mimilakis, and R. Bittner, "MUSDB18-HQ - an uncompressed version of musdb18," Dec. 2019. [Online]. Available: https://doi.org/10.5281/zenodo.3338373