MixingBuddy: A Multimodal LLM for Mix Critique and Advice

Pratham Vadhulas, Alexander Lerch

***Abstract*—Automatic mixing systems based on deep learning have achieved significant progress in producing professional-quality mixes, but they operate as "black boxes" that cannot explain their reasoning or provide interpretable feedback. This limitation hinders their adoption in educational and collaborative music production workflows. We present MixingBuddy, a multimodal large language model that addresses this gap by analyzing multitrack audio and generating structured textual feedback identifying mixing flaws and suggesting corrective gain adjustments. Our architecture employs MERT-v1-330M as an audio encoder, a trainable projection layer to align audio embeddings with the language model's input space, and Qwen2-7B-Instruct as the backbone, fine-tuned with parameter-efficient LoRA adapters. We focus on relative gain relationships among multitrack stems as a tractable starting point for investigating whether multimodal LLMs can learn to reason about mixing balance. On a test set of 10,000 samples, MixingBuddy achieves 50.14% accuracy for the joint task of correctly identifying both the target stem and adjustment direction, with 56.75% stem identification accuracy and 58.41% direction prediction accuracy. When the model correctly identifies the stem and direction, it demonstrates strong magnitude prediction performance (76.04% accuracy). The model shows particularly strong performance on extreme imbalances (78.74% for very loud, 71.35% for very quiet) and exhibits better performance on vocal stems compared to drums and bass, suggesting that semantic understanding provides advantages for certain instrument types. Our work demonstrates that multimodal LLMs can provide interpretable, structured feedback on mixing quality, bridging the gap between automated mixing systems and human understanding.**

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 multimodal large language models (LLMs) bridges that gap. We can now envision a tool that reasons about and discusses a mix.

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).

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 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. 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. [4] introduced Wave-U-Net autoencoders for automatic mixing, demonstrating that end-to-end neural architectures could produce professional-quality mixes. [5] 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 that they are not explainable.

To bridge the semantic gap between audio processing and human understanding, [6] demonstrated word embeddings for automatic equalization in audio mixing, while [7] developed Text2FX, which harnesses CLAP [8] embeddings for text-guided audio effects. Early semantic mixing approaches [9] laid the groundwork for understanding how high-level, semantic knowledge could inform mixing decisions.

Building on language-audio integration, recent work has explored prompt-driven interfaces that map natural language instructions directly to mixing tasks. [10] investigated whether large language models can predict audio effects parameters from natural language, while [11] developed SonicMaster, a controllable all-in-one music restoration and mastering system. [12] introduced MixAssist, an audio-language dataset for co-creative AI assistance in music mixing, demonstrating the potential for collaborative human-AI mixing workflows.

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), converts 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 [13]'s AudioPaLM, [14]'s LauraGPT, and [15]'s SpeechGPT. The unified approach offers the advantage of strong alignment between audio and text. However, this approach is resource intensive and requires LLM pretraining.

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 [16]'s M^2UGen and [17]'s Listen, Think, and Understand (LTU). This approach relies on the theory that aligning audio and text features is sufficient for cross-modal reasoning. The alignment is significatnly simpler than the unified approach.

Our work addresses this gap by leveraging multimodal audio-language models to provide linguistic feedback and reasoning about mixes.

III. Methodology

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 [18], 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) training and evaluation. The dataset comes with a default split into training and test sets of 100 and 50 tracks respectively. 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. Though, during training, we only use the anchor in the instruction as text description, not as a separate audio input. 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: *very quiet*, *quiet*, *no error*, *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 perturb the target stem by applying a gain adjustment within predefined decibel ranges: very quiet (-12 to -6 dB), quiet (-6 to -3 dB), no error (0 dB), loud ($+3$ to $+6$ dB), and very loud ($+6$ to $+12$ dB). The flawed mix is then created by summing all stems, including the modified target stem. Each flawed mix is also augmented during training by applying a random gain adjustment within a predefined range of -3 to $+3$ dB and adding a DC offset and random noise (from -30 to 60 dB) to the audio randomly. This prevents the model from overfitting to the specific audio features of the flawed mixes.

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 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. 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. Similar to the response templates, instructions are generated by randomly selecting from a pool of 10 semantically equivalent instruction templates that vary in wording. The instructions contain context about the mix i.e.the available stems and the anchor stem and a direct instruction to the model to provide gain balancing advice.

This synthesis pipeline produces a large-scale dataset of approximately 100,000 (instruction, response) pairs. The dataset diversity stems from multiple augmentation dimensions: For

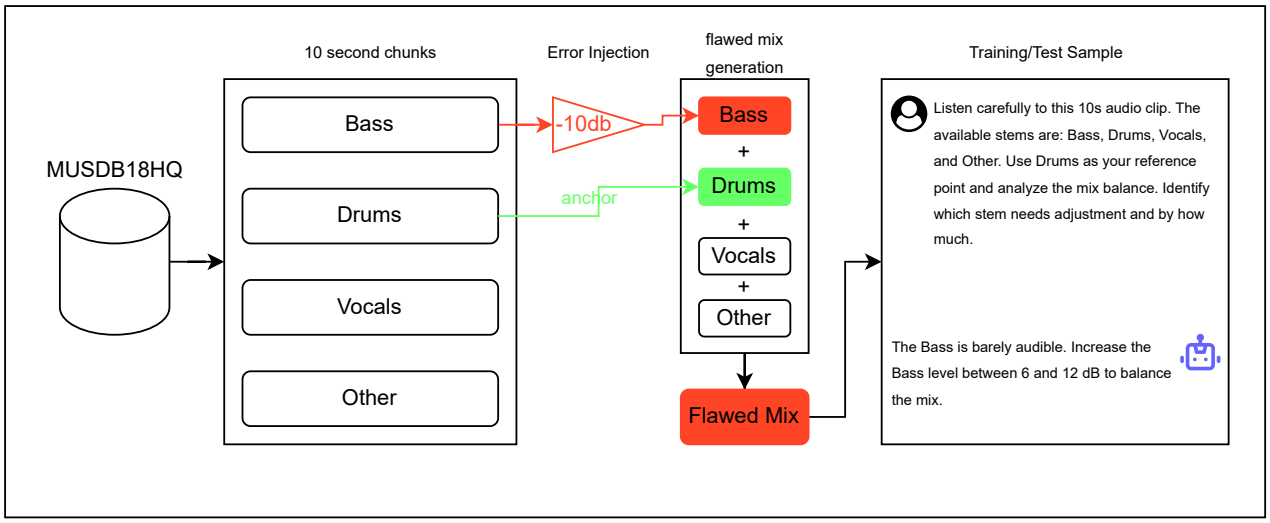

Fig. 1. Overview of the training and test sample synthesis pipeline. The process involves random flaw selection, ground truth label generation from MUSDB18, response template selection, and placeholder population to create synthesized responses.

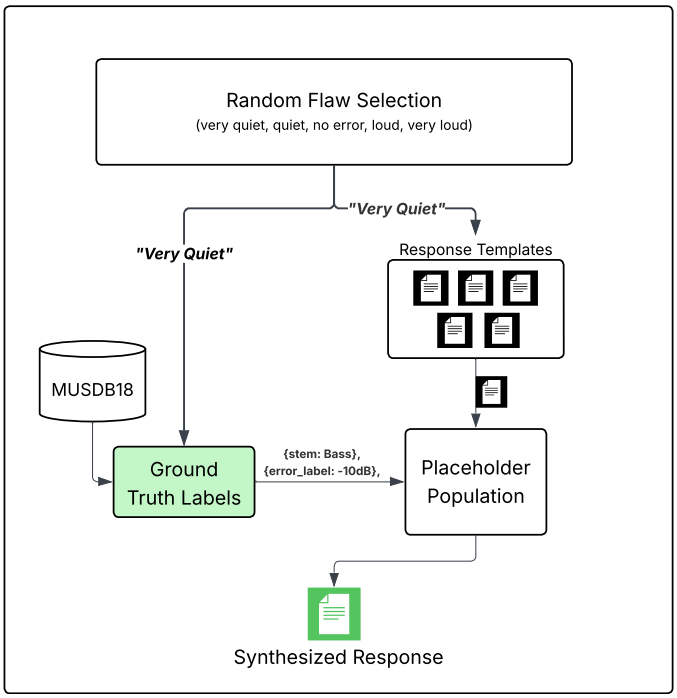

Fig. 2. Response synthesis process. A flaw category is randomly selected, which determines the response template pool. The selected template is then populated with ground truth information including the target stem name and gain adjustment values.

each 10 second chunk, we randomly select 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.

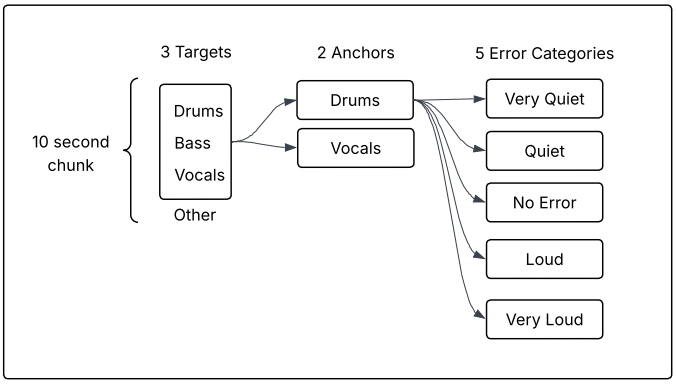

Fig. 3. Dataset augmentation strategy showing the combination of target stems, anchor stems, and error categories that contribute to the final dataset of approximately 100,000 samples (including training and test sets).

B. Architecture

Our architecture implements a multimodal framework that processes 10-second audio segments paired with textual instructions to generate accurate text feedback addressing gain balancing anomalies. Following the audio prefixing approach demonstrated in multimodal language models [17], 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) [17]. While alternative modality alignment techniques exist [16], audio prefixing provides an effective starting point for our application. The architecture employs MERT-v1-330M as the audio encoder [19], 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 [19] as our audio feature extractor. The encoder produces

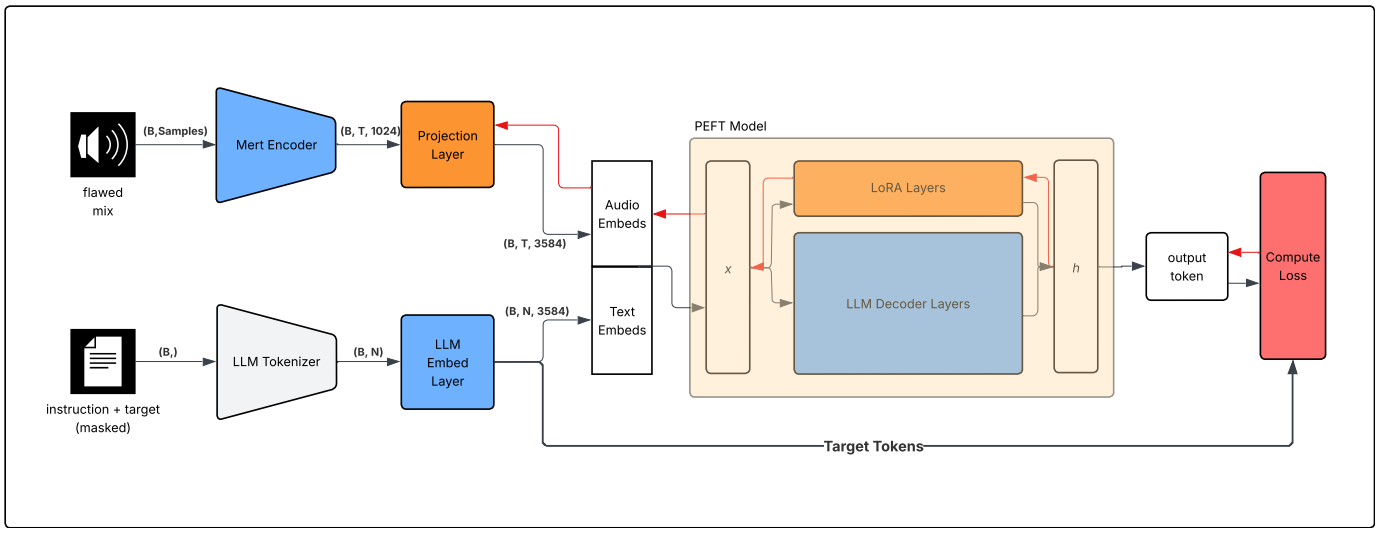

Fig. 4. Architecture overview showing the multimodal framework with audio encoder, projection layer, and language model components.

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 model on the synthesized instruction-response pairs. Our model architecture combines a frozen Qwen2-7B-Instruct backbone with 4-bit quantization, a frozen MERT-v1-330M audio encoder, a trainable MLP projection network, and Low-Rank Adaptation (LoRA) adapters on all linear layers of the language model.

The projection network consists of four hidden layers with dimensions [2048, 4096, 4096, 2048], using GELU activation functions, layer normalization, and dropout regularization. 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 $r = 16$, scaling factor $\alpha = 32$, and dropout rate of 0.1. This configuration allows the language model to adapt its internal representations to better process the audio-conditioned inputs while maintaining parameter efficiency.

1) Implementation Details: The model is implemented using PyTorch [20] with the HuggingFace Transformers library [21] and the Parameter-Efficient Fine-Tuning (PEFT) framework [22]. To enable efficient training of large language models, we employ 4-bit NormalFloat (NF4) quantization [23] with double quantization and bfloat16 compute type through the bitsandbytes library [24]. The audio encoder remains frozen throughout training, while the projection layer and LoRA adapters are trained end-to-end.

2) Hyperparameters: The model is trained using the AdamW optimizer [25] 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, facilitating effective learning of the audio-to-text mapping.

IV. Evaluation

We evaluate the model configuration described in Section III-C to assess its performance on the mixing advice task. We employ a pattern-matching based evaluation approach to assess the accuracy of generated mixing advice.

The evaluation process analyzes each generated response to identify three key components: (1) the target stem that requires adjustment, (2) the direction of the required adjustment (increase, decrease, or no adjustment needed), and (3) the

magnitude of the adjustment expressed as a dB range (e.g., 3–6 dB or 6–12 dB).

1) Accuracy Metrics: We evaluate model performance using four 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.
- **Magnitude accuracy**: The percentage of predictions where the identified adjustment magnitude range (e.g., 3–6 dB or 6–12 dB) matches the expected range based on the error category.

Magnitude accuracy is only evaluated for samples where an adjustment is required (i.e., excluding the "no error" category), as these samples do not have an expected magnitude range.

2) False Positives and False Negatives: In addition to accuracy metrics, we analyze false positives and false negatives to understand the types of errors the model makes:

- **Error detection**: We track false positives (flagging a problem in an error-free mix) and false negatives (missing an actual mixing error).
- **Stem identification**: We track when the model selects the wrong stem or fails to detect the correct one.
- **Direction**: We track when the model predicts the wrong adjustment direction or misses when a direction is needed.

These error type analyses provide insights into the model's failure modes and help identify whether the model tends to be overly conservative (missing errors) or overly aggressive (flagging non-existent problems), which is crucial for practical deployment in mixing workflows.

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).

V. Results

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

A. Projection + LoRA Configuration

We evaluate the configuration employing a trainable projection layer with LoRA adapters on all linear layers of the language model. This configuration allows the model to adapt its internal representations to better process audio-conditioned inputs while maintaining parameter efficiency.

1) Overall Performance: On a test set of 10,000 samples, the projection + LoRA configuration achieves 50.14% accuracy for the joint task of correctly identifying both the target stem and adjustment direction. The model demonstrates 56.75% accuracy for stem identification and 58.41% accuracy for direction prediction. For magnitude prediction, which evaluates whether the model correctly identifies the adjustment range (e.g., 3–6 dB or 6–12 dB), the model achieves 76.04% accuracy on the 8,015 samples where magnitude evaluation is applicable. These results indicate that while the model can identify problematic stems with reasonable accuracy, the joint task of correctly identifying both components remains challenging. However, when the model correctly identifies the stem and direction, it demonstrates strong performance in predicting the appropriate magnitude of adjustment.

2) False Positives, False Negatives, and F1 Scores: Table I presents the false positive and false negative rates, along with precision, recall, and F1 scores for error detection, stem identification, and direction prediction. The model demonstrates strong recall across all tasks (89.99% for error detection, 90.27% for stem identification, 84.69% for direction prediction), indicating that it rarely misses actual mixing problems or fails to identify stems when errors exist. However, precision varies more substantially, with error detection achieving 81.83% precision, stem identification achieving 74.35% precision, and direction prediction achieving 64.47% precision. The F1 scores, which balance precision and recall, reflect this trade-off: error detection achieves the highest F1 score (85.72%), followed by stem identification (81.54%) and direction prediction (73.21%). This suggests that while the model is effective at detecting problems, it sometimes incorrectly flags well-balanced mixes as problematic (1,602 false positives out of 1,985 no-error cases, representing 80.7% of no-error cases). Direction prediction shows the highest false positive rate (3,132 false positives) and the lowest F1 score (73.21%), indicating that the model frequently predicts the wrong adjustment direction, which is particularly concerning as incorrect direction predictions would worsen mix balance.

3) Performance by Error Category: Table II presents the performance breakdown across the five error categories. The model shows strong performance on the *very loud* category, achieving 78.74% accuracy for both stem and direction identification, suggesting that extreme loudness issues are more readily detectable. Magnitude prediction is particularly strong for extreme categories, with 97.76% accuracy for *very loud* and 97.56% for *very quiet*, indicating that when the model

TABLE I
FALSE POSITIVES, FALSE NEGATIVES, PRECISION, RECALL, AND F1 SCORES FOR THE PROJECTION + LORA CONFIGURATION.

Task	FP	FN	Precision	Recall	F1
Error Detection	1602	802	81.83%	89.99%	85.72%
Stem Identification	1903	595	74.35%	90.27%	81.54%
Direction Prediction	3132	1027	64.47%	84.69%	73.21%

TABLE II
PERFORMANCE BREAKDOWN BY ERROR CATEGORY FOR THE PROJECTION + LORA CONFIGURATION.

Category	N	Both	Stem	Dir.	Mag.
very loud	2013	78.74%	84.65%	85.05%	97.76%
very quiet	2007	71.35%	74.54%	80.52%	97.56%
quiet	1997	54.48%	62.54%	68.95%	46.57%
loud	1998	37.59%	53.45%	48.95%	62.01%
no error	1985	7.96%	7.96%	7.96%	–
Macro Avg.	–	**50.02%**	**56.63%**	**58.29%**	**75.97%**

TABLE III
PERFORMANCE BREAKDOWN BY TARGET STEM FOR THE PROJECTION + LORA CONFIGURATION.

Stem	N	Both	Stem	Dir.	Mag.
vocals	3376	56.22%	61.37%	66.47%	93.51%
drums	3322	49.82%	54.52%	55.84%	71.42%
bass	3302	44.25%	54.27%	52.76%	62.85%
Macro Avg.	–	**50.10%**	**56.72%**	**58.36%**	**75.93%**

correctly identifies extreme imbalances, it can reliably predict the appropriate adjustment range. Performance on the *very quiet* category is also strong (71.35% both correct), indicating that the model can effectively identify stems that are too quiet. Performance on the *quiet* category is moderate (54.48% both correct), though magnitude prediction drops to 46.57%, suggesting challenges in determining the appropriate adjustment range for moderate quietness issues. The model struggles most with the *loud* category (37.59% both correct, 62.01% magnitude) and the *no error* category (7.96% both correct), indicating challenges in distinguishing moderate imbalances and correctly identifying well-balanced mixes.

4) Performance by Target Stem: Table III shows the performance breakdown across the three target stems. The model demonstrates best performance on vocals (56.22% both correct), followed by drums (49.82% both correct) and bass (44.25% both correct). Magnitude prediction shows a similar pattern, with vocals achieving 93.51% accuracy, significantly higher than drums (71.42%) and bass (62.85%). This suggests that vocal stem identification may benefit from the model's language understanding capabilities, as vocals often contain more semantic content that can be leveraged for identification, and this advantage extends to magnitude prediction as well.

5) Performance by Direction Type: Table IV presents the performance breakdown by adjustment direction. The model shows stronger performance for *increase* predictions (62.94% both correct) compared to *decrease* predictions (58.24% both correct), suggesting that identifying when stems are too quiet may be slightly easier than identifying when they are too loud. However, magnitude prediction is stronger for *decrease* predictions (79.96%) compared to *increase* predictions (72.13%), indicating that when the model correctly identifies that a stem needs to be reduced, it can more accurately predict the appropriate reduction range. The *no error* category remains challenging (7.96% both correct), consistent with the error category analysis, indicating that the model struggles to recognize well-balanced mixes.

TABLE IV
PERFORMANCE BREAKDOWN BY DIRECTION TYPE FOR THE PROJECTION + LORA CONFIGURATION.

Direction	N	Both	Stem	Dir.	Mag.
increase	4004	62.94%	68.56%	74.75%	72.13%
decrease	4011	58.24%	69.11%	67.07%	79.96%
no error	1985	7.96%	7.96%	7.96%	–
Macro Avg.	–	**43.05%**	**48.54%**	**49.93%**	**76.04%**

VI. CONCLUSION

We have presented MixingBuddy, a multimodal large language model that provides structured mixing advice by analyzing multitrack audio and generating textual feedback identifying problematic stems and suggesting gain adjustments. Our approach demonstrates that LLMs can be effectively adapted for audio-conditioned mixing critique tasks through a combination of trainable projection layers and parameter-efficient LoRA adapters [26].

A. Key Findings

Our experimental evaluation reveals several important insights. The model achieves 50.14% accuracy on the joint task of correctly identifying both the target stem and adjustment direction, with individual accuracies of 56.75% for stem identification and 58.41% for direction prediction. When the model correctly identifies the stem and direction, it demonstrates strong performance in magnitude prediction (76.04% accuracy), indicating that the model can reliably determine appropriate adjustment ranges once the core problem is identified.

The model shows strong recall across all tasks (89.99% for error detection, 90.27% for stem identification, 84.69% for direction prediction), suggesting that it rarely misses actual mixing problems. However, precision varies more substantially, with the model sometimes incorrectly flagging well-balanced mixes as problematic, particularly for the "no error" category (7.96% accuracy). This indicates that while the model is effective at detecting problems, it tends to be overly conservative in recognizing well-balanced mixes.

Performance analysis reveals that the model excels at identifying extreme imbalances (78.74% accuracy for "very loud" and 71.35% for "very quiet" categories) but struggles with moderate imbalances and correctly identifying well-balanced mixes. The model also demonstrates better performance on vocal stems (56.22% both correct) compared to drums (49.82%) and bass (44.25%), suggesting that semantic understanding may provide advantages for certain instrument types.

B. Limitations and Future Work

Several limitations of our current approach suggest directions for future research. First, the model's difficulty in

correctly identifying well-balanced mixes (7.96% accuracy on the "no error" category) represents a significant challenge for practical deployment, as false positives could undermine user trust. Future work could explore techniques to better calibrate the model's confidence or incorporate explicit training on balanced mixes.

Second, the model's performance on moderate imbalances (37.59% for "loud" and 54.48% for "quiet" categories) indicates that subtle mixing issues remain challenging. This may require more sophisticated audio representations, larger training datasets, or alternative architectural approaches that better capture relative gain relationships.

Third, while our focus on gain adjustments provides a tractable starting point, real-world mixing involves many additional parameters such as equalization, compression, and spatial effects. Extending the model to handle these additional mixing parameters represents an important direction for future work.

Finally, the current evaluation relies on pattern-matching to extract structured information from free-form text responses. While this approach provides useful insights, future work could explore more structured output formats or evaluate the model's performance in real-world mixing scenarios with professional audio engineers.

C. Contributions

This work contributes to the field of automatic mixing by demonstrating that multimodal LLMs [16], [17] can provide interpretable, structured feedback on mixing quality. Unlike previous "black box" approaches [4], [5], our model generates textual explanations that can help users understand mixing decisions. The parameter-efficient fine-tuning approach using LoRA adapters [26] makes the model accessible for practical deployment, and the evaluation framework provides a foundation for future research in audio-conditioned language models for music production.

As multimodal LLMs continue to advance, we anticipate that models like MixingBuddy will play an increasingly important role in music production workflows, providing both automated assistance and educational feedback to help users develop their mixing skills.

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] 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.

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

[6] 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.

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

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

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

[10] 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.

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

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

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

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

[15] 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.

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

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

[18] 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

[19] Y. Li, R. Yuan, G. Zhang, Y. Ma, X. Chen, H. Yin, C. Lin, A. Ragni, E. Benetos, N. Gyenge, R. Dannenberg, R. Liu, W. Chen, G. Xia, Y. Shi, W. Huang, Y. Guo, and J. Fu, "Mert: Acoustic music understanding model with large-scale self-supervised training," May 2023.

[20] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," https://pytorch.org, 2019.

[21] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz *et al.*, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, 2020, pp. 38–45.

[22] S. Mangrulkar, S. Gugger, L. Debut, Y. Belkada, S. Paul, J. Eisenstein, P. von Platen, S. Patil, and A. Rush, "Peft: State-of-the-art parameter-efficient fine-tuning methods," https://github.com/huggingface/peft, 2022.

[23] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "Qlora: Efficient finetuning of quantized llms," May 2023. [Online]. Available: https://arxiv.org/abs/2305.14314

[24] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer, "Llm.int8(): 8-bit matrix multiplication for transformers at scale," https://github.com/TimDettmers/bitsandbytes, 2022.

[25] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[26] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," Oct. 2021. [Online]. Available: https://arxiv.org/abs/2106.09685