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. We present MixingBuddy-gain, 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 a pretrained audio encoder, a trainable projection layer to align audio embeddings with the language model's input space, and an instruction-tuned large language model as the backbone, fine-tuned with parameter-efficient LoRA adapters. On a test set of 25,650 samples, MixingBuddy achieves 49.85% accuracy for the joint task of correctly identifying both the target stem and adjustment direction. Our work demonstrates that multimodal LLMs can provide interpretable, structured feedback on mixing quality, bridging the gap between automated mixing systems and human understanding. Code and weights are available at https://github.com/rp-bot/MixingBuddy-gain.**

***Index Terms*—Automatic mixing, Large Language Models, Multimodal learning, Multi-track audio, Music production, Audio analysis, Parameter-efficient fine-tuning, LoRA.**

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.

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, including knowledge engineering and machine learning techniques for instrument-specific effects [1]–[3]. 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. [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, controllability, and diversity. 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. [7] developed Text2FX, which harnesses CLAP [8] embeddings for text-guided audio effects.

Building on language-audio integration, recent work has explored prompt-driven interfaces that map natural language instructions directly to mixing tasks. [9] investigated whether large language models can predict audio effects parameters from natural language, while [10] developed SonicMaster, a controllable all-in-one music restoration and mastering system. [11] 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 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 [12]'s AudioPaLM, [13]'s LauraGPT, and [14]'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 [15]'s M^2UGen and [16]'s Listen, Think, and Under-

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).

stand (LTU). This approach relies on the theory that aligning audio and text features is sufficient for cross-modal reasoning. The alignment is significatnly less resource-intensive than the unified approach.

III. Methodology

We propose a multimodal audio-language model based on the cascade architecture 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 [17], 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.

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.

1) 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.

2) Response Synthesis: To generate corresponding textual responses, we employ a category-driven templating approach. 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.

3) 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 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.

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 [16], 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 for MERT [18]) with the LLM input embedding space (3584 dimensions for Qwen2-7B-Instruct) [16]. While alternative modality alignment techniques exist [15], audio prefixing provides an effective starting point for our application. Figure 2 illustrates the complete architecture.

1) Audio Encoder: We employ the MERT-v1-330M encoder [18] 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

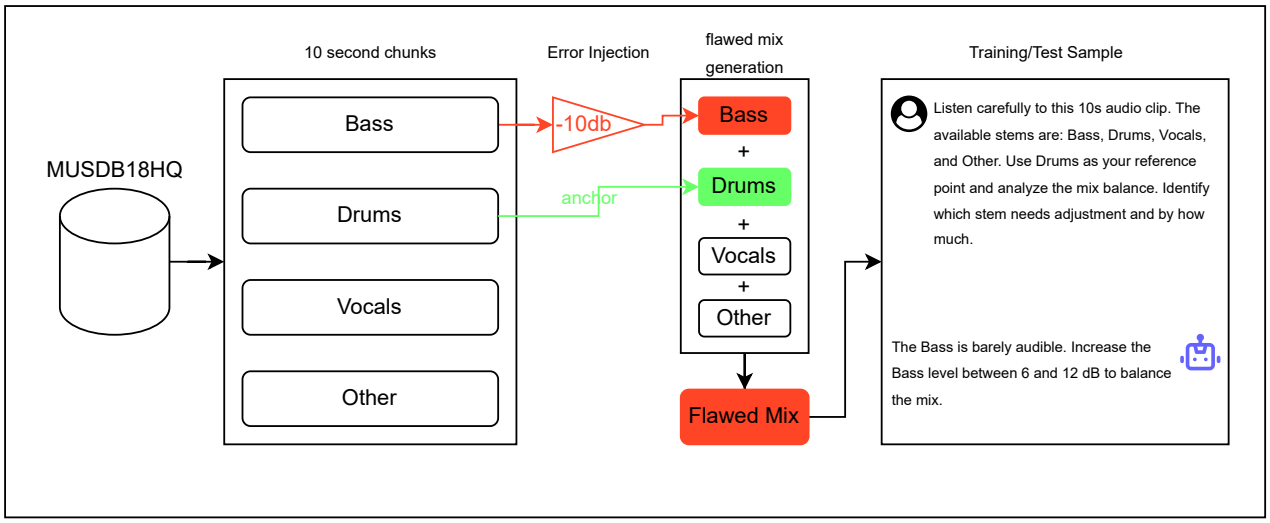

Fig. 1. Overview of the training and test sample synthesis pipeline. The process involves random flaw selection, ground truth label generation from MUSDB18, response template selection, and placeholder population to create synthesized responses.

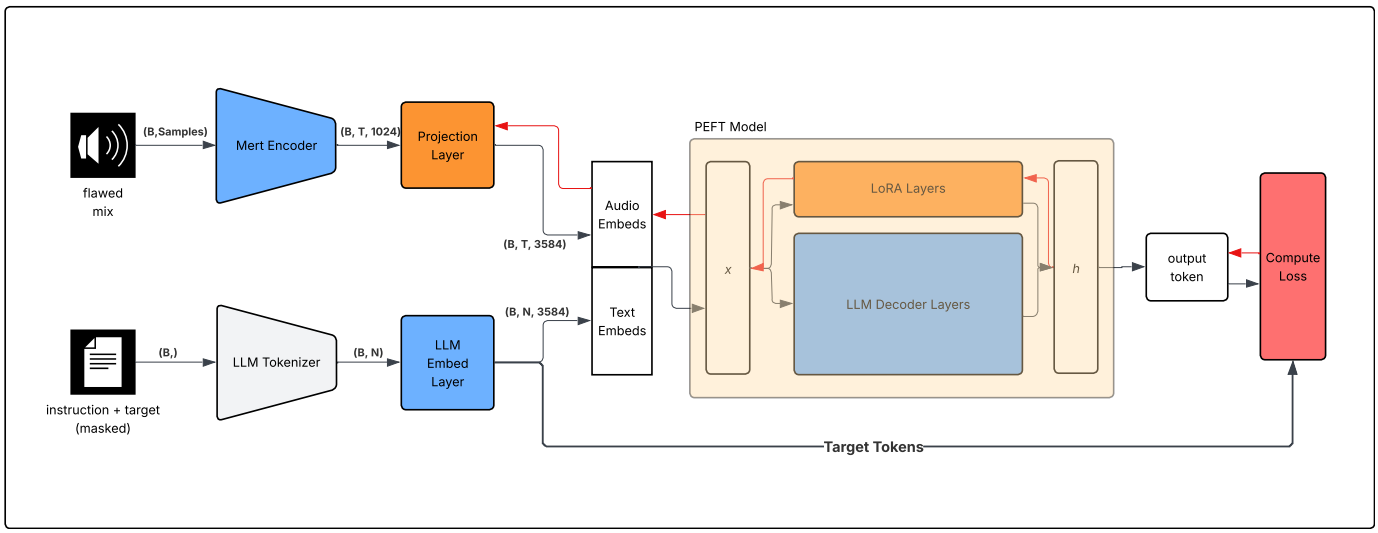

Fig. 2. Architecture overview showing the multimodal framework with audio encoder, projection layer, and language model components with LoRA adapters.

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 with dimensions [2048, 4096, 4096, 2048], 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. The projected audio embeddings are concatenated with text token embeddings and passed to the language model for processing.

3) Language Model: We employ the Qwen2-7B-Instruct language model [19] as the backbone. We train LoRA adapters on the language model to allow fine-tuning while keeping the base model frozen. We apply LoRA 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 allows the language model to both pay attention to these new audio embeddings and generate the advice responses in our expected format.

C. Training Strategy

We employ supervised fine-tuning (SFT) to train our model on the synthesized instruction-response pairs. Our model architecture combines the frozen Qwen2-7B-Instruct backbone with 4-bit quantization, the frozen MERT-v1-330M audio encoder, the 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 using GELU activation functions, layer normalization, and dropout

regularization. LoRA adapters are applied to the attention projection layers and the feed-forward network layers. This configuration allows the language model to adapt its internal representations to better process the audio-conditioned inputs while maintaining parameter efficiency. 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 within a range of ± 0.005 and random noise within a predefined range of -30 to 60 dB to the audio randomly. This prevents the model from overfitting to the specific audio features of the flawed mixes.

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.

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.

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.

IV. Evaluation

We evaluate the model configuration described in Section III-C to assess its performance on the accuracy of the mixing advice task. Our model's advice generation is well aligned with the ground truth advice patterns. This allows us to use the pattern-matching approach to evaluate the accuracy of the generated advice. We use a test set of 25,650 samples to evaluate the accuracy of the generated advice. We evaluate the accuracies of the target stem, direction, and the suggested magnitude range in the generated advice. For each breakdown dimension, we compute both overall performance and performance by error category, stem, and direction.

V. Results

On a test set of 25,650 samples, MixingBuddy-gain achieves 49.85% accuracy for the joint task of correctly identifying both the target stem and error direction (e.g., "The **drums** are **too loud**"). The model achieves 76.06% accuracy for magnitude range prediction on applicable samples.

Table I details the precision, recall, and F1 scores. The model demonstrates consistently strong recall across all tasks ($> 84\%$), indicating it rarely misses actual mixing problems. However, precision varies more substantially, with error detection achieving the highest precision, followed by stem identification and direction prediction. This trade-off is reflected in the F1 scores, where error detection leads, followed by stem identification and direction prediction.

TABLE I
False positives, false negatives, precision, recall, and F1 scores

Task	FP	FN	Precision	Recall	F1
Error Detection	4079	2060	81.9%	89.96%	85.74%
Stem Identification	5035	1545	73.47%	90.02%	80.91%
Direction Prediction	8028	2635	64.38%	84.63%	73.13%

TABLE II
Performance breakdown by error category

Category	N	Both	Stem	Dir.	Mag.
very quiet	5130	69.3%	72.2%	79.9%	97.2%
quiet	5130	53.8%	62.1%	68.8%	46.2%
no error	5130	9.3%	–	–	–
loud	5130	38.1%	52.8%	49.6%	63.2%
very loud	5130	78.8%	84.7%	84.7%	97.6%
Macro Avg.	–	**49.9%**	**56.2%**	**58.4%**	**76.1%**

TABLE III
Performance breakdown by target stem

Stem	N	Both	Stem	Dir.	Mag.
vocals	8550	55.0%	60.3%	65.4%	93.1%
drums	8550	49.9%	54.3%	56.2%	72.3%
bass	8550	44.7%	54.0%	53.6%	62.8%
Macro Avg.	–	**49.9%**	**56.2%**	**58.4%**	**76.1%**

TABLE IV
Performance breakdown by direction type

Direction	N	Both	Stem	Dir.	Mag.
decrease	10260	58.43%	68.75%	67.12%	80.4%
no error	5130	9.3%	–	–	–
increase	10260	61.56%	67.12%	74.32%	71.73%
Macro Avg.	–	**43.09%**	**48.38%**	**50.24%**	**76.06%**

Table II presents the performance breakdown across the five error categories. The model shows strongest performance on extreme categories (*very loud* and *very quiet*), suggesting that significant imbalances are more readily detectable. Magnitude prediction is also particularly reliable for these extreme categories. Performance drops for moderate categories (*loud* and *quiet*), particularly for magnitude prediction in the *quiet* category. The model struggles most with the *no error* category, indicating a challenge in correctly identifying well-balanced mixes without false positives.

Table III shows the performance breakdown across the three target stems. The model performs best on vocals, followed by drums and bass, for both identification and magnitude prediction. 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.

Table IV presents the performance breakdown by adjustment direction. The model shows slightly stronger performance for *increase* predictions compared to *decrease* predictions for identification. However, this trend reverses for magnitude prediction, where *decrease* predictions are more accurate. This suggests that while detecting that a stem is too quiet is easier, determining the precise amount of reduction needed for a loud stem is more reliable once detected.

VI. Conclusion and Future Work

We presented MixingBuddy-gain, a multimodal large language model that provides structured mixing advice by analyzing flawed mixtures and generating textual feedback identifying problematic stems and suggesting gain adjustments. We curated a dataset of approximately 100,000 instruction-response pairs derived from the MUSDB18HQ dataset, employing a synthetic flaw generation pipeline that introduces controlled gain imbalances across bass, drums, and vocal stems. Our methodology integrated a frozen MERT-v1-330M audio encoder with a Qwen2-7B-Instruct language model via a multi-layer perceptron projection network, utilizing audio prefixing to condition the LLM on spectral features. The model was trained using Supervised Fine-Tuning (SFT) with Low-Rank Adaptation (LoRA).

The model achieved promising accuracy on the joint task of correctly identifying both the target stem and adjustment direction, with balanced performance for stem identification and direction prediction. When the model correctly identified the stem and direction, it demonstrated strong performance in magnitude prediction, indicating that the model could reliably determine appropriate adjustment ranges once the core problem was identified. Performance analysis also revealed that the model excelled at identifying extreme imbalances but struggled with moderate imbalances and correctly identifying well-balanced mixes. This indicated that while the model was effective at detecting problems, it sometimes incorrectly flagged well-balanced mixes as problematic. Direction prediction showed the highest false positive rate and the lowest F1 score, indicating that the model frequently predicted the wrong adjustment direction, which is concerning as incorrect direction predictions would worsen mix balance.

These flaws could be attributed to a limitation of the current architecture or the dataset containing inconsistent mixtures. Regarding the architecture, the gradients of the model vanish as it reaches the projection layer (Figure 2). We used an auxiliary loss to enforce alignment between the model's predictions and the audio. But this is merely a workaround and not a true solution. Another potential problem is that currently each time step is projected to the LLM input embedding space. This may not be the most efficient as this creates a tensor of shape $(\text{batch\_size}, \text{time\_steps}, \text{embedding\_size})$ which in our case is $(\text{batch\_size}, 750, 1024)$ before projection and $(\text{batch\_size}, 750, 3584)$ after projection. This may confuse the model as it needs to attend to a lot of information.

Future work could explore a multi step alignment process where a) Train an audio projection layer that is trained to be aligned with the text description of the audio in the LLM's embedding space. b) Use the trained projection layer and replace it in our pipeline (Figure 2) while keeping it frozen. The projection layer can be further strengthened by training it with a contrastive loss objective.

Supervised fine-tuning as shown in this work is not integrated into the huggingface transformers Param Efficient Fine-Tuning (PEFT) pipeline. Hence, we propose it as a framework that can be used as an extension to the PEFT pipeline. We have made the entire codebase available at https://github.com/rp-bot/MixingBuddy-gain. We also provide the weights of the model at insertlink for easy access.

References

[1] B. De Man 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," in *Audio Engineering Society Convention 129*, Nov. 2010.

[3] E. Chourdakis and J. D. 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] D. Koszewski, T. Görne, G. Korvel, and B. Kostek, "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," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Jun. 2021, pp. 71–75.

[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," in *Proceedings of the 27th International Conference on Digital Audio Effects (DAFx24)*, Sep. 2024, pp. 250–257.

[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] 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?" in *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Apr. 2024, pp. 1266–1270.

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

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

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

[13] Z. Du, J. Wang, Q. Chen, Y. Chu, Z. Gao *et al.*, "Lauragpt: Listen, attend, understand, and regenerate audio with gpt," Jul. 2023.

[14] 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," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, Dec. 2023, pp. 15 757–15 773.

[15] 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. 2023.

[16] Y. Gong, H. Luo, A. H. Liu, L. Karlinsky, and J. Glass, "Listen, think, and understand," in *The Twelfth International Conference on Learning Representations*, May 2024. [Online]. Available: https://openreview.net/forum?id=N0StyT5qSj

[17] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, and R. Bittner, "MUSDB18-HQ - an uncompressed version of musdb18," Dec. 2019.

[18] 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.

[19] A. Yang, B. Yang, B. Hui, B. Zheng, B. Yu, C. Zhou, C. Li, C. Li, D. Liu, F. Huang *et al.*, "Qwen2 technical report," *arXiv preprint arXiv:2407.10671*, 2024.

[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.
[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.