LLM2Fx-Tools: Tool Calling For Music Post-Production

Anonymous authors
Paper under double-blind review

Abstract

This paper introduces LLM2Fx-Tools, a multimodal tool-calling framework that generates executable sequences of audio effects (Fx-chain) for music post-production. LLM2Fx-Tools uses a large language model (LLM) to understand audio inputs, select audio effects types, determine their order, and estimate parameters, guided by chain-of-thought (CoT) planning. We also present LP-Fx, a new instruction-following dataset with structured CoT annotations and tool calls for audio effects modules. Experiments show that LLM2Fx-Tools can infer an Fx-chain and its parameters from pairs of unprocessed and processed audio, enabled by autoregressive sequence modeling, tool calling, and CoT reasoning. We further validate the system in a style transfer setting, where audio effects information is transferred from a reference source and applied to new content. Finally, LLM-as-a-judge evaluation demonstrates that our approach generates appropriate CoT reasoning and responses for music production queries. To our knowledge, this is the first work to apply LLM-based tool calling to audio effects modules, enabling interpretable and controllable music production where users can incorporate their own audio plugins. [1]

1 Introduction

The use of Audio effects (Fx) processing constitutes a fundamental component of modern music post-production, where producers systematically apply sequences of effects (Fx-chain) to transform audio signals and achieve desired sound textures (Zölzer et al., 2002; De Man & Reiss, 2013). In most post-production workflows, the application of audio effects is guided by both technical and creative criteria, requiring a high level of expertise from audio engineers. Estimating the appropriate Fx-chain from unprocessed input audio (dry), processed output audio (wet), or by reverse engineering from both requires extensive domain expertise and often involves iterative manual adjustment.

To address this challenge, automatic Fx-chain estimation has emerged as a promising approach to reduce barrier and labor-intensive aspects of music production. Previous works focus on signal processing-based optimization (Barchiesi & Reiss, 2010; Giannoulis et al., 2013; Ma et al., 2015), gradient-based optimization (Colonel & Reiss, 2021; Lee et al., 2024; Steinmetz et al., 2024; Koo et al., 2025; Lee et al., 2025; Yu et al.), regression methods (Rämö & Välimäki, 2019; Sheng & Fazekas, 2019; Mimilakis et al., 2020; Martínez-Ramírez et al., 2021; Steinmetz et al., 2022; Hayes et al., 2025), and multitask methods (Mitcheltree & Koike; Lee et al., 2023; Take et al., 2024). While these methods demonstrate promising performance, they face several fundamental limitations. First, gradient-based methods require differentiable audio effects modules, limiting their applicability to specific effects. Second, regression and signal processing-based methods operate on fixed, predefined configurations and lack the ability to dynamically select effects and determine their ordering. Furthermore, three approaches lack user-level interpretability, as they provide only Fx-chain without human-readable descriptions or reasoning to explain why such decisions are made.

Meanwhile, recent advances in large language models (Achiam et al., 2023; Grattafiori et al., 2024; Comanici et al., 2025) (LLMs) have introduced powerful capabilities including instruction following (Achiam et al., 2023; Wei et al., 2022a), chain-of-thought reasoning (Wei et al., 2022b), and tool calling (Schick et al., 2023). Chain-of-thought (CoT) enables models to decompose a com-

[1] Demo is available at: `https://llm2fx-tool-paper.github.io/demo`

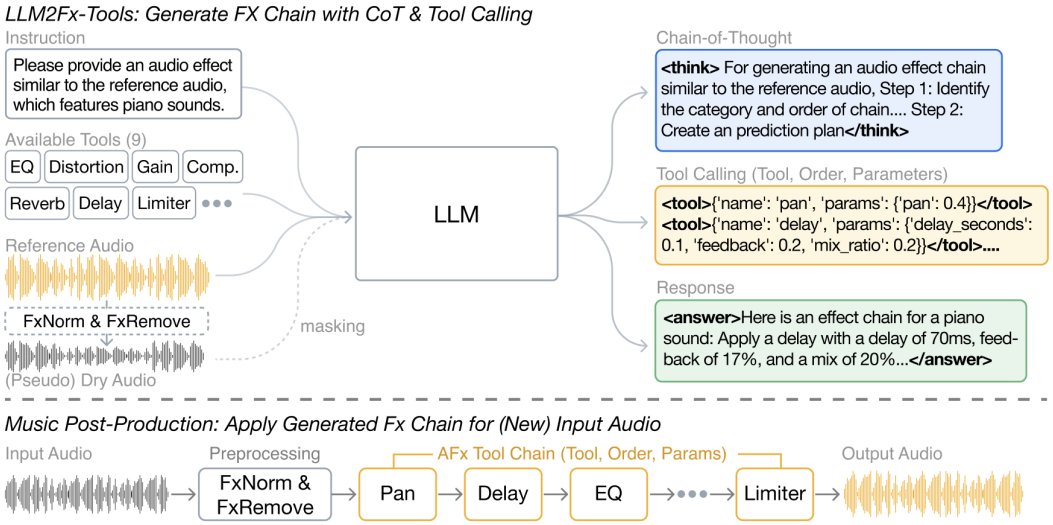

Figure 1: An illustration of the LLM2Fx-Tools framework. The input to LLM2Fx-Tools consists of instruction, available tools, reference audio, and (pseudo) dry audio. The framework outputs chain of thought, tool calling procedure, and response. The generated tool calling outputs (Fx-chain) are then combined with tool environments (audio effects modules) to enable the transformation of new audio in the style of the reference audio.

plex task into a series of reasoning sub-tasks, providing an interpretable view of its reasoning process (Wei et al., 2022b). Tool calling enables LLMs to flexibly connect with external modules (e.g., non-differentiable audio effect modules) and knowledge bases to accomplish domain specific tasks (Schick & Schütze, 2020; Gao et al., 2023). These capabilities present new opportunities to address the flexibility and interpretability issues in Fx-chain prediction. Prior work has explored applying LLMs to music production tasks (Moffat et al., 2022; He et al., 2025). In the context of music post-production specifically, LLM2Fx (Doh et al., 2025) predicts audio effects parameters from natural language prompts, but it does not employ explicit tool calling or chain-of-thought reasoning and is limited to single effects (Equalization and Reverb).

In this work, we introduce LLM2Fx-Tools, LLM-based Fx-chain estimation with Tool-calling, a multimodal framework that addresses these limitations by enabling flexible Fx-chain prediction through tool calling and enhancing interpretability with chain-of-thought reasoning. LLM2Fx-Tools generates 1) chain-of-thought, 2) executable Fx-chain and 3) natural language response. Our key contributions are:

Tool-Calling for Music Production: We develop the first structured tool-calling approach for Fx-chain generation that enables multimodal LLMs to understand audio conditioning and generate executable tool calls for non-differentiable audio effects modules.

Chain-of-Thought for Fx-chain Planning: We utilize a chain-of-thought (CoT) (Wei et al., 2022b) mechanism specifically designed for Fx-chain generation that decomposes the complex task into interpretable sub-tasks: effect selection, order determination, and parameter estimation. This intermediate reasoning bridges the gap between user inputs and target Fx-chain, improving both performance and interpretability.

Multimodal Instruction-Following: We extend the Fx-chain estimation task from unimodal audio-to-effects mapping to a multimodal framework incorporating natural language instructions. Users can specify preferred effect types, musical genres, or instrument characteristics, enabling customized Fx-chains that align with specific user requirements.

Conversational Music Production Dataset: We introduce LP-Fx, LLM-based music production dataset for audio effects tools, containing 101K conversational examples with structured Tool Calling, Chain-of-Thought, and Response. Each example comprises 1) user instructions, 2) audio effects tool calls, 3) chain-of-thought, and 4) responses. Our multi-stage data synthetic methodology

bridges audio signals, Fx-chain, and natural language, establishing a foundation for training multimodal LLMs in music production domains.

2 LLM2Fx-Tools: Fx-chain Generation via Tool Calling

2.1 Task Definition

Our main task is to estimate the Fx-chain ($\mathcal{C}$) that transforms a dry unprocessed audio (x_{dry}) into a reference audio signal (x_{ref}). Formally, given x_{dry} and x_{ref}, our goal is to estimate $\mathcal{C}$ such that $x_{\text{ref}} = \mathcal{E}(\mathcal{C}, x_{\text{dry}})$, where $\mathcal{E}$ denotes the tool environment that applies $\mathcal{C}$ to x_{dry}. For additional controllability, we incorporate natural language instructions ($x_{\text{instruction}}$) to guide the generation process. Our goal is to learn the inverse mapping

$$\mathcal{C} = f_\theta(x_{\text{instruction}}, x_{\text{dry}}, x_{\text{ref}}; \mathcal{T}) \tag{1}$$

where f_θ represents an LLM that predicts the Fx-chain $\mathcal{C} = [(\text{tool}_n, \text{params}_n)]_{n=1}^{N}$ from the reference audio x_{ref} given the available tool set $\mathcal{T}$. We treat each audio effect module tool $\in \mathcal{T}$ as an external executable tool.

As implied in Eq. (1), we consider Fx-chain estimation task where both x_{dry} and x_{ref} are available, which is commonly referred to as the *reverse engineering* (Lee et al., 2025) task. However, x_{dry} is not always accessible in practical scenarios, corresponding to the *blind estimation* (Lee et al., 2023) task. While our primary target is the reverse engineering, we propose a robust training method to simultaneously handle both tasks within a single model, as detailed in Section 2.3.

Our secondary task involves generating chain-of-thought (x_{cot}) and natural language responses (x_{response}). The chain-of-thought (CoT) reasoning serves as an intermediate planning stage that decomposes the complex Fx-chain generation into four sequential components: 1) user input analysis, 2) audio effects module selection, 3) processing order determination, and 4) parameter planning. In our auto-regressive generation framework, the CoT functions as an in-context condition (Wei et al., 2022b) for subsequent tool calling, bridging user queries and action plans to support more accurate and interpretable tool execution. Following the tool calling generation, the model produces natural language responses that provide users with a conversational interface for music production tasks.

2.2 Architecture

To enable LLMs to comprehend audio inputs for tool calling, we adopt a multimodal autoregressive generation framework (Liu et al., 2023; Gardner et al., 2023). As illustrated in Figure 2, since LLMs inherently lack audio processing capabilities, we bridge this modality gap through a pretrained audio encoder coupled with a learnable audio-language adapter. This adapter projects audio representations into the language model's embedding space, formally defined as: $e_{\text{audio}} = f_{\text{adapter}}(f_{\text{encoder}}(x_{\text{audio}}))$, where $f_{\text{encoder}} : \mathbb{R}^{c \times t} \to \mathbb{R}^{l \times d_{\text{enc}}}$ processes input audio signals x_{audio} (c channels, t samples) to extract l audio representations of dimension d_{enc}, and $f_{\text{adapter}} : \mathbb{R}^{l \times d_{\text{enc}}} \to \mathbb{R}^{l \times d_{\text{LLM}}}$ maps these representations to the language model's embedding space of dimension d_{LLM}. The audio embeddings e_{audio} are concatenated with text token embeddings to form a unified multimodal input sequence, enabling the LLM to generate structured outputs including chain-of-thought reasoning x_{cot}, tool calling sequences $\mathcal{C}$, and natural language responses x_{response}.

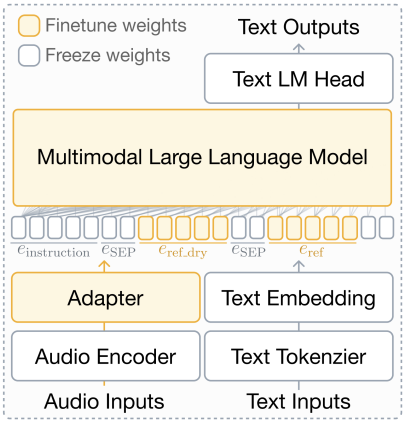

Figure 2: Model Architecture

Audio Encoder: We employ Fx-Encoder++ (Yeh et al., 2025), a specialized audio encoder trained with contrastive learning to obtain representations for audio effects processing. This encoder consists of a ConvNet-based feature extractor, pooling layer, and MLP projection layer. We remove the pooling and MLP projection layers and use the patch embeddings $h_{\text{audio}} \in \mathbb{R}^{l \times d_{\text{enc}}}$ as audio representations.

Adapter: Unlike previous work that uses a simple linear projection layer for cross-modal alignment (Liu et al., 2023; Gardner et al., 2023), we employ a transformer-based audio-language

adapter (Li et al., 2023) with a linear projection layer $W \in \mathbb{R}^{d_{\text{enc}} \times d_{\text{LLM}}}$ and that utilizes 32 learnable query embeddings $e_{\text{query}} \in \mathbb{R}^{32 \times d_{\text{LLM}}}$. This design uses cross-attention to aggregate audio information into learnable query tokens.

Large Language Model: We employ Qwen3-4B (Yang et al., 2025) as our foundation LLM backbone, which provides inherent capabilities for structured tool calling and chain-of-thought reasoning. We fine-tune the model using Low-Rank Adaptation (LoRA) (Hu et al., 2022) with rank 128 and alpha 256 to efficiently adapt the model to our Fx-chain estimation task.

2.3 TRAINING

We employ a unified autoregressive next-token prediction objective to train our multimodal LLM.

Cross-Entropy for Next-token Prediction: Given a training sample with user instruction $x_{\text{instruction}}$, reference audio x_{ref}, dry audio x_{dry}, chain-of-thought x_{cot}, tool calling sequence $\mathcal{C}$, and assistant response x_{response}, we construct the input sequence as a concatenation of a conditioning prefix and a target sequence to be generated by the model as follows:

$$x_{\text{input}} = \underbrace{[x_{\text{instruction}}, x_{\text{dry}}, x_{\text{ref}},}_{\text{Conditioning Prefix}} \underbrace{x_{\text{cot}}, \mathcal{C}, x_{\text{response}}]}_{\text{Target Sequence}} \tag{2}$$

We train the model with the cross-entropy loss $\mathcal{L}_{\text{CE}}$, computed only over the target sequence, while leaving prefix as a conditioning context as follows:

$$\mathcal{L}_{\text{CE}} = -\sum_{t \in T_{\text{target}}} \log p(x_t | x_{<t}; \theta) \tag{3}$$

where T_{target} represents the set of token indices of the target sequence.

Number Token Loss: For parameter estimation, we need to predict the numerical values of effect parameters. However, a key problem with Cross Entropy is that it treats all incorrect predictions equally, even when some numbers are closer to the correct answer than others. In response, we adopt a regression-like Number Token Loss (NTL) that Wasserstein-1 distance between predicted and one-hot number distributions (Zausinger et al., 2025):

$$\mathcal{L}_{\text{NTL-WAS}} = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=s}^{t} \hat{y}_j^i |y_i - V_j| \tag{4}$$

where N is the number of samples, $\hat{y}_j^i$ is the predicted probability for token j in sample i, index s, t representing the range of number token, y_i is the ground truth numerical value, and V_j is the numerical value of token j. This loss function penalizes predictions based on how far they are from the true numerical value, rather than treating all incorrect tokens equally. Our final loss function combines both objectives: $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{CE}} + \lambda \mathcal{L}_{\text{NTL}}$ where λ is a hyperparameter for balancing cross-entropy and number token losses.

Multi-Stage Training: To effectively train our multimodal LLM, we adopt a multi-stage training strategy (Liu et al., 2023) that systematically builds capabilities from basic audio-language alignment to complex reasoning tasks. Our training protocol comprises two distinct phases: 1) modality alignment pre-training and 2) LLM fine-tuning while progressively incorporating task complexity.

We first pre-train the adapter module to bridge the audio modality and text modality. During the pre-training stage, we use only audio inputs and tool calling outputs (Fx-chain) as training data, focusing solely on learning the relationship between dry and reference audio differences and their corresponding Fx-chain. In this stage, we freeze the LLM parameters and only update the audio-language adapter parameters. In the fine-tuning stage, we update both the adapter and LLM through LoRA adaptation. This stage incorporates the full complexity of our task, including not only tool calling but also chain-of-thought reasoning and response generation.

Robust Training Techniques for Distribution Shift: As mentioned in Section 2.1, our goal is to estimate $\mathcal{C}$ such that $x_{\text{ref}} = \mathcal{E}(\mathcal{C}, x_{\text{dry}})$, assuming we have access to both x_{dry} and x_{ref}. However, training a model only on a dataset of paired (x_{dry}, x_{ref}) audio samples for this *reverse engineering* setup, real-world scenarios present distribution shift challenges: x_{dry} is typically unavailable during inference. Even when it is available, it may differ from the training dataset since recording studios vary significantly in their equipment and environments.

To address this challenge, we employ Fx-Normalization (Martínez-Ramírez et al., 2022) and Fx-Removal (Rice et al., 2023) techniques at both training and inference stages to align dry audio distributions and obtain pseudo dry audio $\hat{x}_{\text{dry}}$. Furthermore, we implement dry audio masking during training, randomly omitting dry audio inputs with probability p_{masking} to force the model to rely solely on reference audio for the *blind estimation* setup.

3 Dataset: LP-Fx

We propose a novel LLM-based data synthesis pipeline that systematically generates high-quality conversation data for Fx-chain generation tasks, introducing a new dataset as one of the key contributions of this work.

3.1 Base Dataset and Tool Environment

The audio source of LP-Fx is MedleyDB (Bittner et al., 2014; 2016), which provides royalty-free 196 multitrack recordings. Each recording includes three different levels of audio: (i) unprocessed *raw* tracks, (ii) *stems*, which are submixes of raw tracks with audio effects applied, and (iii) a full *mix*, created by combining the processed stems into a complete mixture. We use unprocessed raw audio as the dry audio x_{dry}. We filter out multitracks with bleed using the metadata provided by MedleyDB, resulting in a curated set of 2,119 raw audio files from 116 multitracks, spanning 9 genres and 80 unique instruments. We use the `Pedalboard` [2] audio effects library and our custom audio effects modules as our tool environment $\mathcal{T}$. We select 6 modules (compressor, distortion, reverb, delay, limiter, and gain) from the Pedalboard library and 3 modules (three-band equalizer, stereo widener, and panner) from our custom modules, totaling 9 modules and 26 parameters.

3.2 Data Generation Process

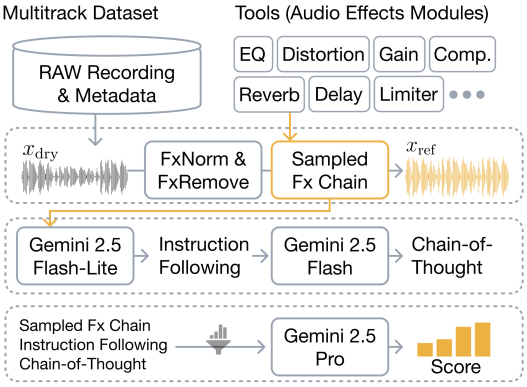

Figure 3: Data generation process for LP-Fx

As illustrated in Figure 3, our data generation process consists of three sequential stages. In the first stage, we sample Fx-chains within musically plausible ranges to create dry/processed audio pairs. The second stage generates instruction-following conversations grounded in these Fx-chains to ensure factual accuracy. The third stage produces chain-of-thought reasoning that explicitly connects user instructions to the underlying audio effects transformations. Finally, we employ LLM-as-a-judge (Chen et al., 2024; Zheng et al., 2023) evaluation to filter the dataset for high-quality samples. The basic framework for utilizing LLMs in our data generation process follows the formulation $x_{\text{data}} = \text{LLM}(x_{\text{ground}}, \mathcal{P})$, where x_{ground} is the grounded information (e.g., Fx-chain $\mathcal{C}$) and $\mathcal{P}$ is the carefully crafted prompts. We further provide details of each stage in Appendix B.

3.3 Statistics

Table 1 presents the statistics of our LP-Fx dataset. Based on the observation that task complexity increases with the number of effects in the chain, we create a stratified dataset with 11,100 training samples and 100 test samples for each Fx-chain length from 1 to 9, resulting in a total of 100,800 conversations. We ensure no overlap between audio tracks used in training and test sets to prevent data leakage. Each example comprises 1) user instructions, 2) unprocessed/processed audio pairs, 3) executable audio

Table 1: Statistics of LP-Fx Dataset.

Statistics	Trainset	Testset
# of Dialogue	99,900	900
# of Tracks	2019	100
# of Instruments	80	33
Avg. Instruction length	28.8	28.2
Avg. Response length	178.6	178,6
Avg. CoT length	251.9	252.7
Min/Max Tool Number	1-9	1-9

[2] `https://github.com/spotify/pedalboard`

effects tool calls, 4) chain-of-thought reasoning, and 5) assistant responses. The dataset's rich diversity in musical content makes it particularly effective for LLM fine-tuning. With 2,019 tracks spanning 80 unique instruments across 9 genres.

4 EXPERIMENTS

4.1 REVERSE ENGINEERING

Task Definition: We evaluate our approach on *reverse engineering*. The task involves predicting Fx-chains $\mathcal{C}$ from reference audio x_{ref} given access to the corresponding dry audio x_{dry}. We evaluate this task using the LP-Fx test split, which provides ground truth triplets of $(x_{\text{dry}}, x_{\text{ref}}, \mathcal{C})$ for evaluation.

Metrics: Our evaluation framework assesses model performance through four complementary perspectives: 1) *Fx-chain Planning*, 2) *Perceptual Distance*, 3) *DSP Feature Distance*, and 4) *Deep Embedding Similarity*. For *Fx-chain Planning*, we use classification accuracy to measure whether the model correctly predicts the presence of target audio effects modules in the ground truth Fx-chain. We then apply Spearman rank correlation to assess how closely the predicted ordering of modules matches the ground truth. Finally, parameter mean absolute error (MAE) is used to quantify the precision of parameter prediction. For *Perceptual Distance*, we employ Multi-Resolution STFT (MRS) distance (Yamamoto et al., 2020) on both left-right (L/R) and mid-side (M/S) channels for stereo-aware processing evaluation. For *DSP Feature Distance*, we utilize Audio Features (AF) (Man et al., 2014; Vanka et al., 2024), including root mean square, crest factor, stereo width, stereo imbalance, and bark spectrum. For *Deep Embedding Similarity*, we employ audio effects-specific pretrained representations, including classification-based AFx-Rep (Steinmetz et al., 2024) and contrastive learning-based Fx-Encoder (Koo et al., 2023).

Baselines: We evaluate our approach against several baselines to assess the effectiveness of our instruction-following Fx-chain generation framework. *1) No Fx:* A naive baseline that applies no audio effects to the input audio, representing the lower bound performance where the predicted reference audio $\hat{x}_{\text{ref}}$ equals the dry audio x_{dry}. *2) Random Fx:* A baseline that applies the random number of effects with randomized ordering and parameters. *3) Regression:* A regression-based approach that directly predicts audio effects parameters from audio features without explicit tool selection or ordering capabilities. Specifically, we first extract embeddings using the Fx-Encoder++ Yeh et al. (2025), followed by a 2-layer MLP with ReLU activations. The regression head outputs a vector of logits corresponding to the number of parameters in the full Fx-chain. *4) Multitask:* An enhanced regression model incorporating additional classification heads to address limitations of the pure regression approach. The base architecture is identical to the regression model, but additional logits are predicted to classify which audio effects modules are applied. *5) Gemini 2.5 Flash* (Comanici et al., 2025): A closed-source multimodal LLM with audio understanding, reasoning, and tool calling capabilities. The regression and multitask baselines employ the same pretrained audio encoder as LLM2Fx-Tools, MLP projection layers and task-specific prediction heads.

4.2 AUDIO EFFECTS STYLE TRANSFER WITH BLIND ESTIMATION

Task Definition: We evaluate our approach on the audio effects style transfer task, which simulates real-world scenarios where users only have access to a reference audio with different content. This task comprises two sequential stages: 1) *blind estimation* - inferring the underlying Fx-chain $\mathcal{C}$ from a processed reference audio x_{ref} without the knowledge of its corresponding original dry recording, and 2) *style transfer* - applying the estimated Fx-chain to another source audio. This evaluation assesses the model's ability to generalize to unseen musical content.

Evaluation Protocol: To evaluate generalization across different musical content, we employ MoisesDB (Pereira et al., 2023) as the source of processed reference stems and testset of LP-Fx (Bittner et al., 2014; 2016) as the source of clean input audio. This cross-dataset evaluation protocol ensures that models encounter entirely unseen musical content, providing a rigorous test of generalization capabilities. We construct evaluation pairs by matching instrument categories between the two datasets, resulting in an evaluation set of 100 test samples. Given that the reference and input audio contain distinct musical content, we focus our evaluation on feature-based metrics, including DSP feature distance and embedding similarity. We employ the same baseline methods of the *reverse engineering task* as described in 4.1.

Table 2: Fx-chain Estimation Results. We compare with multiple baselines and analyze the contribution of key components in our LLM2Fx-Tools framework: Chain-of-Thought (CoT), Number Token Loss (NTL), and Multi-Stage Training (MST).

	Fx-chain Planning			Perceptual Dist.		DSP	Embedding Sim.(↑)	
	Acc.(↑)	Corr.(↑)	MAE(↓)	L/R(↓)	M/S(↓)	AF(↓)	AFx-Rep	FxEnc
No Fx	-	-	-	13.11	13.49	14.82	0.50	0.30
Random Fx	52%	-0.01	0.39	8.07	8.90	13.70	0.41	0.34
Regression	55%	-0.03	**0.20**	3.81	4.12	9.20	0.62	0.64
MultiTask	61%	0.00	0.23	3.17	3.39	8.39	0.63	0.66
$\text{Gemini2.5}_{\text{Flash}}$	78%	0.54	0.32	3.42	4.24	14.97	0.56	0.50
LLM2Fx-Tools	**80%**	**0.56**	0.23	**3.13**	**3.27**	**8.29**	**0.68**	**0.67**
w/o CoT	67%	0.49	0.24	3.34	3.38	8.39	0.64	0.66
w/o NTL	73%	0.51	0.32	3.69	3.52	8.47	0.61	0.63
w/o MST	76%	0.55	0.25	3.21	3.32	8.30	0.67	0.64

4.3 Natural Language Generation

Task Definition: Beyond the Fx-chain estimation capabilities, LLM2Fx-Tools generates chain-of-thought reasoning and natural language responses, providing interpretability and transparency to users through comprehensive explanations of the audio processing decisions.

Evaluation Protocol: We evaluate the natural language generation quality of our LLM2Fx-Tools framework. Following previous works (Gardner et al., 2023; Clemens & Marasović, 2025), we assess the natural language generation capabilities through an LLM-as-a-judge framework (Zheng et al., 2023). Similar to Section 3.2, we use Gemini2.5-Pro model as $\text{LLM}_{\text{judge}}$. Specifically, we evaluate three key dimensions: 1) *tool calling success*, whether the model correctly executes the required Fx-chain, 2) *instruction following quality*, whether the generated response adequately addresses the user instruction, and 3) *chain-of-thought quality*, whether the reasoning effectively connects user instructions to responses through coherent intermediate steps. This process can be formulated as $(s_{\text{IF}}, s_{\text{CoT}}) = \text{LLM}_{\text{judge}}(x_{\text{instruction}}, x_{\text{response}}, x_{\text{cot}}, \mathcal{P}_{judge})$. $\mathcal{P}_{judge}$ details provided in Appendix E.

Baselines: We compare our approach with LLMs for natural language generation: *1) Qwen2.5-Omni 7B:* (Chu et al., 2024) An open-source multimodal LLM without reasoning capabilities, *2) Qwen 2.5 4B:* (Yang et al., 2025) A compact open-source LLM without audio understanding, and *3) Gemini 2.5 Flash:* (Comanici et al., 2025) A closed-source multimodal LLM with advanced reasoning capabilities.

4.4 Training / Evaluation Details

We utilize Qwen3-4B (Yang et al., 2025) as our pretrained LLM foundation, which provides instruction following, reasoning and tool calling ability. Training is performed across multi-stage training (MST) with different learning rates and batch sizes. For Stage 1 (modality alignment pretraining), we use a learning rate of 1e-4 with batch size of 32 and train for 100K steps. Stage 2 (LLM finetuning) employs a learning rate of 5e-5 with batch size of 16 and is iterated for 400K steps.

5 Results / Analysis

5.1 Reverse Engineering

Comparison on Fx-chain Planning. Table 2 demonstrates that LLM2Fx-Tools achieves superior performance across multiple evaluation dimensions. In *Fx-chain Planning*, our approach significantly outperforms all baselines, achieving 80% accuracy in audio effects module classification and 0.56 Spearman correlation for ordering, compared to the MultiTask baseline with 61% accuracy and near-zero correlation. While the regression baseline achieves slightly better parameter MAE (0.20 vs 0.23), this comes at the cost of substantially worse audio effects module selection and ordering capabilities. Interestingly, Gemini 2.5 Flash demonstrates strong *Fx-chain Planning* capabilities with 78% effect classification accuracy and reasonable ordering correlation (0.54). However, it exhibits limitations in parameter estimation, achieving the highest parameter MAE (0.32).

The performance improvements of LLM2Fx-Tools stem from two key aspects: 1) our instruction-following capabilities leverage natural language understanding to provide additional conditioning beyond pure audio comprehension, enabling more precise and semantically-aware audio processing decisions; and 2) the autoregressive sequence modeling inherent in LLMs provides a fundamental advantage for handling Fx-chain ordering compared to models that rely solely on audio features.

Does Fx-chain Planning Lead to Better Acoustic Similarity? For *Perceptual Distance*, LLM2Fx-Tools achieves the best performance on both MRS distances, outperforming all baselines. Our analysis indicates that effective *Fx-chain Planning* is essential not only for accurate parameter prediction but also for achieving strong perceptual performance. Notably, while the regression baseline achieves the lowest parameter MAE (0.20), this advantage in parameter space does not translate into improved perceptual distance. Because the regression model lacks the ability to selectively apply audio effects modules, it must predict parameters for all predefined modules, even when they are absent in the reference audio. This limitation leads to suboptimal perceptual and feature-level reconstruction, underscoring the importance of Fx-chain planning for bridging parameter accuracy and perceptual quality.

In contrast, both the MultiTask baseline and our LLM2Fx-Tools framework, which incorporate audio effects module selection capabilities, demonstrate superior performance in both perceptual and DSP distance compared to the base regression approach. Comparing MultiTask vs LLM2Fx-Tools further demonstrates the critical importance of Fx-chain ordering: despite achieving similar DSP distances (8.39 vs 8.29), LLM2Fx-Tools's substantial improvement in ordering correlation (0.56 vs 0.00) leads to better perceptual reconstruction (3.13 vs 3.17 L/R MRS). This indicates that correct effect sequencing significantly contributes to audio processing quality, as the order of effects can dramatically alter the final audio output. For *Deep Embedding Similarity*, LLM2Fx-Tools achieves the highest similarity scores (AFx-Rep: 0.68, Fx-Encoder: 0.67), demonstrating that effective Fx-chain outputs more semantically similar to reference audio.

Ablation Studies. The lower portion of Table 2 demonstrates that our core design choices contribute meaningfully to model performance. Chain-of-Thought (CoT) reasoning significantly aids Fx-chain Planning capabilities, improving effect classification accuracy from 67% to 80% and enhancing ordering correlation from 0.49 to 0.56. Number Token Loss (NTL) notably impacts parameter estimation, reducing MAE from 0.32 to 0.23, while also improving overall perceptual and feature-level metrics. MST provides improvements across all metrics, bridging the representations between the pretrained audio encoder and LLM while leveraging the pretrained capabilities of Qwen3.

5.2 Audio Effects Style Transfer with Blind Estimation

Table 3 presents the experiment results of audio effects style transfer, designed to evaluate the cross-domain generalization capabilities of each method. We observe similar trends to those seen in reverse engineering experiments. The regression baseline, which applies all predefined audio effects regardless of their relevance, achieves suboptimal performance with higher DSP distance (7.83) and lower embedding similarity scores. The MultiTask approach shows modest improvements with better DSP distance (7.62) and enhanced embedding similarities, highlighting the importance of selective effect application. However, its lack of ordering capabilities limits further performance gains.

Table 3: Audio Effects Style Transfer Results.

	DSP	Embedding Sim.	
	AF(↓)	AFx-Rep(↑)	FxEnc(↑)
No Fx	8.69	0.24	0.43
Random Fx	15.22	0.14	0.19
Regression	7.83	0.24	0.31
MultiTask	7.62	0.29	0.46
Gemini2.5$_{\text{Flash}}$	9.00	0.24	0.27
LLM2Fx-Tools	**7.41**	**0.35**	**0.49**

Among LLM-based approaches, Gemini 2.5 Flash performs poorly, yielding a DSP distance of 9.00 and embedding similarities barely above the No Fx baseline, despite its large parameter count. Our reverse engineering experiments reveal that its parameter predictions are nearly random, which explains why it fails to generalize effectively to the style transfer task. In contrast, LLM2Fx-Tools consistently achieves the best results, with the lowest DSP distance (7.41) and the highest embedding similarity scores (AFx-Rep: 0.35, Fx-Encoder: 0.49) across all evaluated methods.

Table 4: Natural Language Generation Results. We compare with multiple (multimodal) large language models on tool calling (TC), instruction following (IF), and chain of thought (CoT).

	Params	Multimodal	Reasoning	TC Success	IF Quality	CoT Quality
Qwen 2.5_{Omni}	7B	✓	✗	0.2%	2.01	N/A
Qwen 3	4B	✗	✓	73.7%	3.63	3.62
Gemini 2.5_{Flash}	N/A	✓	✓	**100%**	3.75	3.63
LLM2Fx-Tools	4B	✓	✓	99.8%	**3.86**	**3.69**

5.3 Natural Language Generation

Table 4 evaluates natural language generation capabilities through comprehensive LLM-as-a-Judge assessment. Qwen 2.5_{Omni} demonstrates limited zero-shot tool calling capabilities, failing to generate correctly formatted JSON structures. In contrast, Qwen 3-4B achieves substantial tool calling success (73.7%) despite lacking multimodal capabilities, indicating the effectiveness of text-based reasoning for this task. LLM2Fx-Tools, built upon Qwen 3-4B with specialized multimodal training, achieves near-perfect tool calling performance (99.8%), matching the capabilities of state-of-the-art closed-source models such as Gemini 2.5 Flash (100.0%).

The instruction following evaluation reveals substantial quality improvements from domain-specific training. LLM2Fx-Tools outperforms zero-shot baselines, with quality scores increasing from 3.63 to 3.86 compared to the base Qwen 3 model. This improvement demonstrates the critical importance of specialized training for music production tasks, where general-purpose models lack domain-specific knowledge about audio effects and their applications.

Additionally, CoT quality evaluation shows that LLM2Fx-Tools outperforms other LLM baselines. This can be attributed to LLM2Fx-Tools's training on high-quality CoT dataset specifically focused on music production tasks. The training dataset LP-Fx leverages knowledge distillation from Gemini2.5 Flash, with quality assurance provided through filtering by Gemini2.5 Pro. This multi-stage data synthesis approach is expected to enhance the CoT capabilities of the fine-tuned model, enabling more coherent and domain-relevant reasoning for audio effects manipulation.

6 Limitation

While our framework advances interpretable and controllable Fx-chain estimation, several challenges remain. First, the predicted Fx-chain is interpretable only relative to pseudo-dry audio obtained through Fx-normalization and Fx-removal preprocessing. Full interpretability would require integrating these preprocessing steps directly into the Fx-chain representation and reasoning process. Second, the inherent one-to-many mapping in audio effects estimation (Hayes et al., 2025) creates ambiguity where multiple distinct Fx-chains can produce perceptually similar results, particularly in symmetric parameter spaces. Our current evaluation framework does not fully account for this fundamental ambiguity. Third, our experimental validation focuses exclusively on single-instrument sources, which may limit direct applicability to complex multitrack music production scenarios. Furthermore, we have not evaluated generalization to unseen audio effects modules beyond our training distribution. While our tool-calling framework is designed to be extensible to new VST plugins and audio effects modules, empirical validation of this capability remains future work.

7 Conclusion

We present LLM2Fx-Tools, a multimodal tool-calling framework for generating executable audio effect chains with interpretable chain-of-thought planning. Our experimental results demonstrate that LLM2Fx-Tools outperforms regression and multitask baselines on both reverse engineering and audio style transfer tasks. Additional evaluations through LLM-as-a-judge confirm the natural language generation capabilities of our approach, demonstrating strong instruction following and chain-of-thought reasoning quality. By emitting structured tool calls over audio effect modules, our approach enables controllable and explainable music post-production applications. This work opens several promising directions for future research, including scaling to richer VST plugin toolboxes, developing reinforcement learning frameworks, and conducting comprehensive perceptual evaluations with expert music producers to validate real-world applicability.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Daniele Barchiesi and Joshua Reiss. Reverse engineering of a mix. *Journal of the Audio Engineering Society*, 58(7/8):563–576, 2010.

Rachel M Bittner, Justin Salamon, Mike Tierney, Matthias Mauch, Chris Cannam, and Juan Pablo Bello. MedleyDB: A multitrack dataset for annotation-intensive mir research. In *The 15th International Society for Music Information Retrieval Conference (ISMIR)*, volume 14, pp. 155–160, 2014.

Rachel M Bittner, Julia Wilkins, Hanna Yip, and Juan P Bello. MedleyDB 2.0: New data and a system for sustainable data collection. *ISMIR Late Breaking and Demo Papers*, 36, 2016.

Dongping Chen, Ruoxi Chen, Shilin Zhang, Yaochen Wang, Yinuo Liu, Huichi Zhou, Qihui Zhang, Yao Wan, Pan Zhou, and Lichao Sun. MLLM-as-a-judge: Assessing multimodal llm-as-a-judge with vision-language benchmark. In *Forty-first International Conference on Machine Learning*, 2024.

Yunfei Chu, Jin Xu, Qian Yang, Haojie Wei, Xipin Wei, Zhifang Guo, Yichong Leng, Yuanjun Lv, Jinzheng He, Junyang Lin, et al. Qwen2-audio technical report. *arXiv preprint arXiv:2407.10759*, 2024.

Michael Clemens and Ana Marasović. Mixassist: An audio-language dataset for co-creative ai assistance in music mixing. *arXiv preprint arXiv:2507.06329*, 2025.

Joseph T Colonel and Joshua Reiss. Reverse engineering of a recording mix with differentiable digital signal processing. *The Journal of the Acoustical Society of America*, 150(1):608–619, 2021.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Brecht De Man. *Towards a better understanding of mix engineering*. PhD thesis, Queen Mary University of London, 2017.

Brecht De Man and Joshua D Reiss. A knowledge-engineered autonomous mixing system. In *Audio Engineering Society Convention 135*. Audio Engineering Society, 2013.

Seungheon Doh, Junghyun Koo, Marco A Martínez-Ramírez, Wei-Hsiang Liao, Juhan Nam, and Yuki Mitsufuji. Can large language models predict audio effects parameters from natural language? In *2025 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2025.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: Program-aided language models. In *International Conference on Machine Learning*, pp. 10764–10799. PMLR, 2023.

Josh Gardner, Simon Durand, Daniel Stoller, and Rachel M Bittner. LLark: A multimodal instruction-following language model for music. *arXiv preprint arXiv:2310.07160*, 2023.

Dimitrios Giannoulis, Michael Massberg, and Joshua D Reiss. Parameter automation in a dynamic range compressor. *Journal of the Audio Engineering Society*, 61(10):716–726, 2013.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Ben Hayes, Charalampos Saitis, and GyĂśrgy Fazekas. Audio synthesizer inversion in symmetric parameter spaces with approximately equivariant flow matching. In *The 26th International Society for Music Information Retrieval Conference (ISMIR)*, 2025.

Qi He, Gus Xia, and Ziyu Wang. Tomi: Transforming and organizing music ideas for multi-track compositions with full-song structure. In *The 26th International Society for Music Information Retrieval Conference (ISMIR)*, 2025.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. LoRA: Low-rank adaptation of large language models. *International Conference on Learning Representations*, 2022.

Junghyun Koo, Marco A Martínez-Ramírez, Wei-Hsiang Liao, Stefan Uhlich, Kyogu Lee, and Yuki Mitsufuji. Music mixing style transfer: A contrastive learning approach to disentangle audio effects. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023.

Junghyun Koo, Marco A Martinez-Ramirez, Wei-Hsiang Liao, Giorgio Fabbro, Michele Mancusi, and Yuki Mitsufuji. Ito-master: Inference-time optimization for audio effects modeling of music mastering processors. In *The 26th International Society for Music Information Retrieval Conference (ISMIR)*, 2025.

Sungho Lee, Jaehyun Park, Seungryeol Paik, and Kyogu Lee. Blind estimation of audio processing graph. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2023.

Sungho Lee, Marco A Martínez-Ramírez, Wei-Hsiang Liao, Stefan Uhlich, Giorgio Fabbro, Kyogu Lee, and Yuki Mitsufuji. Searching for music mixing graphs: A pruning approach. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, 2024.

Sungho Lee, Marco A Martínez-Ramírez, Liao Wei-Hsiang, Stefan Uhlich, Giorgio Fabbro, Kyogu Lee, and Yuki Mitsufuji. Reverse engineering of music mixing graphs with differentiable processors and iterative pruning. *AES: Journal of the Audio Engineering Society*, 73(6):344–365, 2025.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pp. 19730–19742. PMLR, 2023.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.

Zheng Ma, Brecht De Man, Pedro DL Pestana, Dawn AA Black, and Joshua D Reiss. Intelligent multitrack dynamic range compression. *Journal of the Audio Engineering Society*, 63(6):412–426, 2015.

BD Man, Brett Leonard, Richard King, Joshua D Reiss, et al. An analysis and evaluation of audio features for multitrack music mixtures. In *The 15th International Society for Music Information Retrieval Conference (ISMIR)*, 2014.

Marco A Martínez-Ramírez, Oliver Wang, Paris Smaragdis, and Nicholas J Bryan. Differentiable signal processing with black-box audio effects. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021.

Marco A Martínez-Ramírez, Wei-Hsiang Liao, Giorgio Fabbro, Stefan Uhlich, Chihiro Nagashima, and Yuki Mitsufuji. Automatic music mixing with deep learning and out-of-domain data. In *The 23rd International Society for Music Information Retrieval Conference (ISMIR)*, 2022.

Stylianos I Mimilakis, Nicholas J Bryan, and Paris Smaragdis. One-shot parametric audio production style transfer with application to frequency equalization. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 256–260. IEEE, 2020.

Christopher Mitcheltree and Hideki Koike. White-box audio vst effect programming. *NeurIPS 2020 Workshop on Machine Learning for Creativity and Design*.

David Moffat, Brecht De Man, and Joshua D Reiss. Semantic music production: A meta-study. *Journal of the Audio Engineering Society*, 2022.

Igor Pereira, Felipe Araújo, Filip Korzeniowski, and Richard Vogl. Moisesdb: A dataset for source separation beyond 4-stems. *arXiv preprint arXiv:2307.15913*, 2023.

Pedro Duarte Leal Gomes Pestana. *Automatic mixing systems using adaptive digital audio effects*. Phd thesis, Universidade Catolica Portuguesa, 2013.

Jussi Rämö and Vesa Välimäki. Neural third-octave graphic equalizer. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*. University of Birmingham, 2019.

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

Timo Schick and Hinrich Schütze. Exploiting cloze questions for few shot text classification and natural language inference. *arXiv preprint arXiv:2001.07676*, 2020.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551, 2023.

Di Sheng and György Fazekas. A feature learning siamese model for intelligent control of the dynamic range compressor. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2019.

Christian J Steinmetz, Nicholas J Bryan, and Joshua D Reiss. Style transfer of audio effects with differentiable signal processing. *Journal of the Audio Engineering Society*, 70(9):708–721, 2022.

Christian J Steinmetz, Shubhr Singh, Marco Comunità, Ilias Ibnyahya, Shanxin Yuan, Emmanouil Benetos, and Joshua D Reiss. ST-ITO: Controlling audio effects for style transfer with inference-time optimization. In *The 25th International Society for Music Information Retrieval Conference (ISMIR)*, 2024.

Osamu Take, Kento Watanabe, Takayuki Nakatsuka, Tian Cheng, Tomoyasu Nakano, Masataka Goto, Shinnosuke Takamichi, and Hiroshi Saruwatari. Audio effect chain estimation and dry signal recovery from multi-effect-processed musical signals. In *Proc. Int. Conf. Digital Audio Effects (DAFx)*, pp. 1–8, 2024.

Soumya Sai Vanka, Christian Steinmetz, Jean-Baptiste Rolland, Joshua Reiss, and George Fazekas. Diff-MST: Differentiable mixing style transfer. In *The 25th International Society for Music Information Retrieval Conference (ISMIR)*, 2024.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022a.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022b.

Ryuichi Yamamoto, Eunwoo Song, and Jae-Min Kim. Parallel WaveGAN: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6199–6203. IEEE, 2020.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Yen-Tung Yeh, Junghyun Koo, Marco A Martínez-Ramírez, Wei-Hsiang Liao, Yi-Hsuan Yang, and Yuki Mitsufuji. Fx-Encoder++: Extracting instrument-wise audio effects representations from mixtures. In *The 26th International Society for Music Information Retrieval Conference (ISMIR)*, 2025.

Chin-Yun Yu, Marco A Martínez-Ramírez, Junghyun Koo, Wei-Hsiang Liao, Yuki Mitsufuji, and György Fazekas. Improving inference-time optimisation for vocal effects style transfer with a gaussian prior. *2025 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*.

Jonas Zausinger, Lars Pennig, Anamarija Kozina, Sean Sdahl, Julian Sikora, Adrian Dendorfer, Timofey Kuznetsov, Mohamad Hagog, Nina Wiedemann, Kacper Chlodny, et al. Regress, don't guess–a regression-like loss on number tokens for language models. In *International Conference on Machine Learning*, 2025.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging LLM-as-a-judge with MT-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

Udo Zölzer, Xavier Amatriain, Daniel Arfib, Jordi Bonada, Giovanni De Poli, Pierre Dutilleux, Gianpaolo Evangelista, Florian Keiler, Alex Loscos, Davide Rocchesso, et al. *DAFX-Digital audio effects*. John Wiley & Sons, 2002.

A Detailed Task Definition

Borrowing notation from Rice et al. (2023), we define $\mathcal{T}$ as the set of effect functions supported by the environment ($\mathcal{E}$). Given a dry audio (x_{dry}), a typical audio processing can be formulated as a composition of functions that yields the processed audio (x_{wet}) as follows:

$$x_{wet} = g_N\Big(g_{N-1}\Big(\cdots g_2\big(g_1(x_{dry};\phi_1);\phi_2\big)\cdots;\phi_{N-1}\Big);\phi_N\Big), \tag{5}$$

where $g_n \in \mathcal{T}$ is the n-th effect function and ϕ_n denotes the parameters of g_n. In this paper, we assume $g_i \neq g_j$ for any pair of $i \neq j$ for simplicity. We denote the Fx-chain by $\mathcal{C} = [(g_n, \phi_n)]_{n=1}^N$ = $[(\text{tool}_n, \text{params}_n)]_{n=1}^N$. Equation 5 can be reformulated in terms of $\mathcal{E}$ and $\mathcal{C}$ as $x_{wet} = \mathcal{E}(\mathcal{C}, x_{dry})$.

Our main task is to reverse-engineer the Fx-chain ($\mathcal{C}$) applied to a reference audio signal (x_{ref}). Specifically, given a processed reference audio signal x_{ref}, we aim to predict the sequence of audio effects and their parameters that were used to create the processed version from an original dry signal (x_{dry}). Formally, we can express this relationship as $x_{\text{ref}} = \mathcal{E}(\mathcal{C}, x_{\text{dry}})$, where the environment ($\mathcal{E}$) applies the Fx-chain ($\mathcal{C}$) to the dry audio (x_{dry}) to produce the processed reference audio (x_{ref}). For additional controllability, we incorporate natural language instructions ($x_{\text{instruction}}$) to guide the generation process. Our goal is to learn the inverse mapping

$$\hat{\mathcal{C}} = f_\theta\Big(x_{\text{instruction}}, x_{\text{dry}}, x_{\text{ref}}; \mathcal{T}\Big), \tag{6}$$

where f_θ represents an LLM that aims to estimate the original Fx-chain $\mathcal{C} = [(\text{tool}_n, \text{params}_n)]_{n=1}^N$ from the reference audio x_{ref} and x_{dry} with an additional input $x_{\text{instruction}}$ in the provided environment $\mathcal{T}$.

Our secondary task involves generating intermediate chain-of-thought (x_{cot}) and natural language responses (x_{response}). The chain-of-thought reasoning serves as an intermediate planning stage that decomposes the complex Fx-chain generation into four sequential components: 1) user input analysis, 2) effect selection, 3) processing order determination, and 4) parameter planning. In our autoregressive generation framework, the chain-of-thought functions as an in-context condition for subsequent tool calling, bridging user queries and action plans to support more accurate and interpretable tool execution. Following the tool calling generation, the model produces natural language responses that provide users with a conversational interface for music production tasks.

B Detailed Dataset Generation

We detail out each stage of the data generation pipeline for creating LP-Fx below as mentioned in Section 3.2.

Stage 1: Dry/processed audio pairs synthesis. For synthesizing processed reference audio x_{ref}, we apply the sampled Fx-chain $\mathcal{C}$ to the dry audio x_{dry} from MedleyDB. We apply Fx-Normalization (in the order of EQ, stereo imager, and loudness) and Fx-Removal to the dry audio samples and create a normalized dry audio $\hat{x}_{\text{dry}}$. We randomly sample parameters within predefined min-max ranges and quantize them to discrete steps that mirror practical knob granularity (Pestana, 2013). We employ two sampling regimes: a coarse regime to broadly cover the operating space and a fine regime, which reflect real world production practices (De Man, 2017) (sampling ranges are detailed in Table 5). Consequently, we obtain $(\hat{x}_{\text{dry}}, x_{\text{ref}}, \mathcal{C})$ triplets where each triplet contains the original dry audio, the processed reference audio, and the corresponding Fx-chain sequence.

Stage 2: Instruction-following synthesis. We synthesize natural single-turn conversations between users and assistants for music production scenarios using the Fx-chains generated in Stage 1. For efficient large-scale generation, we employ a distillation LLM, Gemini-2.5-Flash-lite (Comanici et al., 2025). In this stage, the Fx-chain sequence $\mathcal{C} = [(\text{tool}_n, \text{params}_n)]$ from Stage 1 is paired with task prompts $\mathcal{P}_{chat}$ that describe realistic music production scenarios. The LLM then generates natural language instructions $x_{\text{instruction}}$ and assistant responses $x_{\text{responses}}$ that preserve the underlying Fx-chain structure while providing contextually appropriate explanations, formally expressed as $x_{\text{instruction}}, x_{\text{response}} = \text{LLM}(x_{\text{tool}}, \mathcal{P}_{chat})$.

Stage 3: Chain-of-thought generation. To bridge the gap between the Fx-chains $\mathcal{C}$ generated in Stage 1 and the instruction-response pairs $x_{\text{instruction}}, x_{\text{response}}$ from Stage 2, we decompose the music production task into a step-by-step manner. We construct chain-of-thought reasoning by dividing the tool calling process into four sequential steps: 1) user input analysis, 2) tool selection, 3) ordering, and 4) parameter planning. We utilize Gemini-2.5-Flash (Comanici et al., 2025) with enhanced reasoning capabilities for this stage. This process can be formulated as $x_{\text{cot}} = \text{LLM}(x_{\text{instruction}}, x_{\text{response}}, \mathcal{C}, \mathcal{P}_{cot})$ where $\mathcal{P}_{cot}$ represents the task prompts that guide the decomposition of complex audio processing into interpretable reasoning steps.

Stage 4: Quality filtering. To ensure data quality and minimize hallucinations in our synthetic dataset, we employ an LLM-as-a-judge evaluation framework using Gemini 2.5 Pro (Comanici et al., 2025). We implement a two dimensional quality assessment evaluating: 1) *tool alignment*, whether generated conversations accurately align with the grounded tool information, Fx-chain $\mathcal{C}$, and 2) *CoT quality*, whether chain of thought reasoning effectively guides from user queries to tool calling sequences. This process can be formulated as $(s_{\text{tool}}, s_{\text{CoT}}) = \text{LLM}_{\text{judge}}(x_{\text{instruction}}, x_{\text{response}}, x_{\text{cot}}, \mathcal{C}, \mathcal{P}_{judge})$ where s_{tool} and s_{CoT} are quality scores of *tool alignment* and *CoT*, respectively, and $\mathcal{P}_{judge}$ is the evaluation prompts. Both s_{tool} and s_{CoT} are evaluated using a 4 point Likert scale (from 1=poor to 4=excellent), where samples scoring ≤ 2 are flagged for regeneration to maintain dataset integrity. $\mathcal{P}_{judge}$ details provided in Appendix E.

C Qualitative Examples

In this section, we present a qualitative example below illustrating how LLM2Fx-Tools processes a user query and generates an appropriate Fx-chain. The example highlights each stage of the system's reasoning and output: the user's request, the model's chain-of-thought for Fx-chain modeling and parameter estimation, the corresponding tool calls, and the final response presented to the user.

User Query
I'm working on a track with Electronic/Fusion synthesizer sounds, and I'd like to emulate the audio effects style of a reference audio. Could you suggest a plugin chain with a stereo widener, distortion, and a compressor?

Chain-of-Thought
For emulating the audio effects style of a reference audio for Electronic/Fusion synthesizer sounds using a stereo widener, distortion, and compressor:
Step 1. The identified audio effects in the chain are stereo_widener, distortion, and compressor, in that specific order.
Step 2. For stereo_widener, the width parameter will be adjusted to create an expansive and immersive stereo image, enhancing the synth's presence.
For distortion, the drive_db will be increased to introduce harmonic richness and saturation, adding grit and character suitable for electronic fusion.
For compressor, the threshold_db will be set to effectively manage the dynamic range of the synth, the ratio will be chosen to provide controlled dynamic reduction, the attack_ms will be adjusted to either preserve or shape the initial transients, and the release_ms will be set to allow the compressor to disengage smoothly with the synth's decay, ensuring a natural sound.

Tool Calling
"<tool_call>\n{'name': 'stereo_widener', 'arguments': {'width': 1.3}}\n</tool_call>
<tool_call>\n{'name': 'distortion', 'arguments': {'drive_db': 5.0}}\n</tool_call>
<tool_call>\n{'name': 'compressor', 'arguments': {'threshold_db': -19.0, 'ratio': 4.0, 'attack_ms': 250.0, 'release_ms': 250.0}}\n</tool_call>

Response
Here is a plugin chain designed to emulate the audio effects style of your reference audio, featuring a stereo widener, distortion, and a compressor. The stereo widener is set to a width of 1.3 to expand the stereo image. Following this, a distortion effect with a drive of 5.0 dB is applied to add harmonic content.
Finally, a compressor with a threshold of -19.0 dB, a ratio of 4.0, an attack of 250.0 ms, and a release of 250.0 ms is used for dynamic control.",

D Evaluation Metric

Audio Effects Module Classification Accuracy: We evaluate the model's ability to correctly identify which audio effects module should be applied using standard classification accuracy:

$$\text{Acc} = \frac{\text{correct predictions}}{\text{total predictions}}. \tag{7}$$

Fx-Chain Order Correlation: We assess Fx-chain ordering (order of audio effects module) capability using Spearman rank correlation ρ between predicted and ground truth orders. Missing values are set to $|\text{fx\_pool}| + 1$ for consistent ranking evaluation.

Audio Effects Parameter MAE: We calculate Mean Absolute Error for parameter prediction: $\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|\hat{p}_i - p_i|$, where parameters are normalized to $[0, 1]$ before computation. This metric only considers correctly classified effects.

Left/Right MRS: Multi-Resolution STFT distance computed separately for stereo channels: $\text{MRS} = \sum_k (\mathcal{L}_{\text{sc}}^{(k)}(\hat{x}_{\text{ref}}, x_{\text{ref}}) + \mathcal{L}_{\text{mag}}^{(k)}(\hat{x}_{\text{ref}}, x_{\text{ref}}))$, where k indexes different time-frequency resolutions, and

$$\mathcal{L}_{\text{sc}}^{(k)}(x, \hat{x}) = \frac{\left\| |\text{STFT}^{(k)}(x)| - |\text{STFT}^{(k)}(\hat{x})| \right\|_F}{\left\| |\text{STFT}^{(k)}(x)| \right\|_F}, \tag{8}$$

$$\mathcal{L}_{\text{mag}}^{(k)}(x, \hat{x}) = \frac{1}{N}\left\| \log|\text{STFT}^{(k)}(x)| - \log|\text{STFT}^{(k)}(\hat{x})| \right\|_1. \tag{9}$$

Mid/Side MRS: We convert stereo audio to Mid/Side representation and compute MRS distance. Mid-channel captures mono content (addition of left and right channels) while Side-channel captures stereo width and spatial characteristics (subtraction of left and right channels).

DSP Feature Distance: We extract digital signal processing (DSP) based low-level descriptors, including the root mean square and crest factor, stereo width and stereo imbalance and bark spectrum corresponding to the dynamics, spatialization and spectral attributes respectively.

Embedding Similarity: We use pretrained audio effect encoders to extract semantic representations and compute cosine similarity for different types of audio encoders, including CLAP, AFX-Reps, and Fx-Enc.

E Prompt Details

We present the detailed prompts used for our dataset generation and LLM-as-a-judge evaluation.

Instruction-Following Generation Prompts: We use two main prompts for generating our dataset. The first prompt guides the model to generate realistic user-assistant conversations with appropriate tool calls.

```
You are a post-production assistant (mixing and mastering) specialized in
    audio processing and VST plugin chains.
Complete the following conversation.

Output:
[
    {{
        "role": "user",
        "content": [user_instruction]
    }},
    {{
        "role": "assistant",
        "content": [assistant_response]
    }}
```

```
]

Tools:
{fx_chain}

Requirements:
- User requests audio effect parameters of the reference audio. {
    str_user_instruction} {str_user_request_specific_fx}
- The reference audio contains {genre} {instrument} sounds.
- In the assistant message, please keep tool number {tool_numer} and the
    tool order {tool_order}
- In the assistant message, briefly explain the audio effect type, order
    and parameters with natural language description. Please provide
    objective information, don't use overly subjective words. Please
    answer with a short and concise description.
```

Chain-of-Thought Generation Prompts: The second prompt specifically focuses on generating chain-of-thought reasoning that bridges multimodal understanding with parameter prediction.

```
You are a post-production assistant (mixing and mastering) specialized in
     audio processing and VST plugin chains.
Given a Audio Effects Chain and a previous tool-based chat conversation,
    generate the next chain-of-thought plan.
Return ONLY a single valid JSON object. Do not include any text before or
     after the JSON. Do not use markdown fences.

Outputs:
{{
    "chain_of_thought": "<think>For [task description], Step1,.. Step2,..
      </think>"
}}

Where:
- task description: The task description is the user's request.
- chain_of_thought: A step-by-step explanation that covers:
- Step 1. From the reference audio, identify the category and order of
    audio effects in the chain. Do not specify exact values.
- Step 2. Create an FX parameter prediction plan that describes the
    general direction and approach for each effect's parameters without
    specifying exact values.

Constraints:
- Use the provided Audio Effects Chain for effect and parameter names;
    match names exactly.
- Chain of thought reflects the assistant's thinking process for analysis
     and parameter prediction.

Audio Effects Chain:
{vst_info}

conversations:
{conversation}
```

LLM-as-a-Judge Prompts1: For evaluate dataset generation, we evaluate for tool alignment and thought quality.

```
You are an expert evaluator for audio post-production conversations
    involving VST plugin chains.
Evaluate the assistant's response in the given conversation based on the
    following criteria.

Use scores to show the quality of the response. Here is the detailed
    scoring rubric for evaluating the quality of responses
from AI assistants:
```

```
# Tool Alignment (Order, Direction, Parameter Accuracy):
Poor (1): Significant misalignment with tool chain order, incorrect
    parameter directions, and highly inaccurate parameter values that
    would produce undesirable audio results.
Fair (2): Partial alignment with tool order but contains noticeable
    errors in parameter direction or accuracy.
Good (3): Strong alignment with tool order, correct parameter directions,
     and accurate parameter values with only minor room for improvement.
Excellent (4): Perfect alignment with tool chain order, correct parameter
     directions, and highly accurate parameter values demonstrating
    expert-level understanding.

# Thought Quality:
Poor (1): Illogical chain of thought lacking coherent reasoning about
    audio processing decisions.
Fair (2): Basic reasoning but contains gaps in logic or limited
    understanding of audio processing principles.
Good (3): Strong reasoning with clear understanding of effect
    interactions and good audio processing knowledge.
Excellent (4): Expert-level reasoning with sophisticated understanding of
     complex effect interactions.
{{
    "tool_alignment": {{
        "score": [1, 2, 3, 4],
    }},
    "thought_quality": {{
        "score": [1, 2, 3, 4],
    }},
}}

Tool calling ground truth:
{fx_chain}

Conversation to evaluate:
{conversation}
```

LLM-as-a-Judge Prompts2: For natual langauge generation, we evaluate for instruction following and chain of thought quality.

```
You are an expert evaluator for audio post-production conversations
    involving VST plugin chains.
Evaluate the assistant's response in the given conversation based on the
    following criteria.

Use scores to show the quality of the response. Here is the detailed
    scoring rubric for evaluating the quality of responses
from AI assistants:
# Instruction Following Quality:
Poor (1): The response does not follow the user's instructions, ignores
    key requirements, or provides irrelevant information. The answer is
    not in natural language or does not address the task described in the
     instruction.
Fair (2): The response partially follows the instructions, but misses
    important details or only addresses some aspects of the user's
    request. The natural language answer may be incomplete or only
    loosely related to the instruction.
Good (3): The response follows the instructions well, addresses most
    requirements, and provides a mostly complete and relevant answer in
    natural language that matches the task in the instruction, but may
    lack some detail or completeness.
Excellent (4): The response fully follows the user's instructions,
    addresses all requirements in detail, and provides a clear, relevant,
     and comprehensive answer in natural language that is directly
    aligned with the task described in the instruction.
```

```
# Chain of Thought Quality:
Poor (1): The chain of thought does not logically connect the user's
    query to the assistant's response, lacking coherent reasoning about
    audio processing decisions. The reasoning fails to demonstrate proper
     task decomposition, analysis of user input, and planning for effect
    chain implementation. Or the chain of thought is empty.
Fair (2): The reasoning attempts to bridge the user's query and the
    assistant's response but contains gaps in logic or shows limited
    understanding of audio processing principles. Some evidence of task
    decomposition and planning to handle user input may be present but
    incomplete or flawed.
Good (3): The chain of thought clearly links the user's query to the
    assistant's response, demonstrating effective task decomposition and
    planning. The reasoning provides clear evidence of user input
    analysis and systematic planning to handle requirements with mostly
    sound logic.
Excellent (4): The reasoning expertly bridges the user's query and the
    assistant's response through comprehensive task decomposition and
    strategic planning. The analysis demonstrates thorough task
    decomposition, comprehensive planning to handle user input, and
    expert-level reasoning throughout the process.
{{
    "instruction_following_quality": {{
        "score": [1, 2, 3, 4],
    }},
    "chain_of_thought_quality": {{
        "score": [1, 2, 3, 4],
    }},
}}

Conversation to evaluate:
{conversation}

Chain of thought:
{cot}
```

F PARAMETER RANGE FOR DATASET SAMPLING

Table 5: Parameter space of the audio effects used in this study. For each parameter, we define the range and discretized step size for both coarse and fine-grained search spaces.

	Coarse		Fine	
Parameter	Range	Step	Range	Step
Three-band Equalizer				
low gain db	[-20.0, 20.0]	2	[-6.0, 6.0]	1
low cutoff freq	[0.0, 400.0]	20	[60.0, 120.0]	10
low Q factor	[0.0, 6.0]	0.5	[0.5, 3.0]	0.25
mid gain db	[-20.0, 20.0]	2	[-6.0, 6.0]	1
mid cutoff freq	[250.0, 6000.0]	250	[250.0, 1000.0]	100
mid Q factor	[0.1, 6.0]	0.5	[0.5, 3.0]	0.25
high gain db	[-20.0, 20.0]	2	[-6.0, 6.0]	1
high cutoff freq	[4000.0, 20000.0]	1000	[4000.0, 8000.0]	500
high Q factor	[0.0, 6.0]	0.5	[0.5, 3.0]	0.5
Compressor				
threshold db	[-40.0, -5.0]	5	[-20.0, -10.0]	1
ratio	[0.0, 20.0]	1	[2.0, 8.0]	0.5
attack ms	[0.0, 500.0]	5	[1.0, 30.0]	1
release ms	[0.0, 1000.0]	50	[0.0, 500.0]	25
Stereo Widener				
width	[0.0, 1.5]	0.1	[1.1, 1.5]	0.1
Gain				
gain db	[-20.0, 20.0]	2	[-6.0, 6.0]	1
Panner				
pan	[-1.0, 1.0]	0.1	[-0.6, 0.6]	0.1
Distortion				
drive db	[0.0, 20.0]	2	[1.0, 5.0]	0.5
Reverb				
room size	[0.0, 0.9]	0.1	[0.3, 0.6]	0.05
damping	[0.0, 0.9]	0.1	[0.3, 0.6]	0.05
width	[0.0, 0.9]	0.1	[0.3, 0.6]	0.05
mix ratio	[0.0, 1.0]	0.1	[0.1, 1.0]	0.1
Relay				
delay seconds	[0.0, 0.7]	0.05	[0.01, 0.2]	0.02
feedback	[0.0, 0.6]	0.05	[0.01, 0.2]	0.02
mix ratio	[0.0, 1.0]	0.1	[0.1, 1.0]	0.1
Limiter				
threshold db	[-20.0, -1.0]	1	[-5.0, -1.0]	0.1
release ms	[0.0, 1000.0]	50	[0.0, 300.0]	25