To further bridge the gap between objective metrics and human auditory perception, we assess the models' subjective performance in Table 4. SQ-Codec outperforms its counterparts across all seven subjective dimensions. Notably, in terms of Vocal Similarity (VS) and Melody Naturalness (MN), SQ-Codec achieves scores of 3.90 and 4.25, respectively, significantly higher than 1D VAE. Taking all factors into consideration, we follow the setting of SimpleSpeech Yang et al. [2024, 2025a], and designate the bottleneck features of SQ-Codec as the target latent for the Flow Matching module.

2.3.4 Ablation Study of Training Stages

To further assess the impact of reflow distillation and fine-tuning SQ-Codec on downstream music generation tasks, we conduct an additional set of experiments. Specifically, our music generation model, HeartMuLa (see Section 3.1), is trained to model token sequences produced by HeartCodec. During inference, we let HeartMuLa predict a sequence of tokens given lyrics and style tags as prompts, keep the predicted tokens fixed, and decode them using HeartCodec obtained at different training stages.

Specifically, we consider HeartCodec after pretraining and finetuning (Pt. & Ft.), after reflow distillation (Reflow), and after SQ-Codec fine-tuning (SQ Ft.). Notably, during both reflow distillation and SQ-Codec fine-tuning, the encoder and compressor components of HeartCodec are kept frozen. Therefore, using the same token sequence as input across different decoders is well justified and allows for a controlled comparison.

After decoding, we evaluate the generated music using metrics including aesthetic metrics produced by AudioBox Tjandra et al. [2025], musical quality measured by SongEval Yao et al. [2025], style alignment measured by tag similarity (Tag-Sim) and intelligibility of the vocal track measured by the phoneme error rate (PER). We report results on our HeartBeats-Benchmark (English) dataset (see Sec. 6.5), as summarized in Table 5. As shown in the results, compared to HeartCodec (Pt. & Ft.), the model after reflow distillation achieves noticeable improvements in both aesthetic quality and tag similarity, albeit with a degradation in vocal intelligibility. After further fine-tuning the SQ-Codec , all evaluation metrics consistently improve, yielding the best overall performance among the compared methods. These results demonstrate that the two additional training stages, reflow distillation and SQ-Codec fine-tuning, consistently enhance the quality of downstream music generation.

Table 5: Objective Evaluation of the Effects of Training Stages on Downstream Music Generation.

Model	**AudioBox** $\uparrow$			**SongEval** $\uparrow$						**Align** $\uparrow$	**Intel** $\downarrow$
	CE	CU	PQ	Coh.	Mus.	Mem.	Cla.	Nat.	Avg.	Tag-Sim	PER
HeartCodec (Pt. & Ft.)	7.37	7.78	8.05	4.43	4.24	4.43	4.30	4.13	4.31	0.2339	0.1092
HeartCodec (Reflow)	7.42	7.78	8.04	4.46	4.28	4.46	4.32	4.18	4.34	0.2486	0.1258
HeartCodec (SQ Ft.)	**7.45**	7.78	**8.07**	**4.52**	**4.36**	**4.54**	**4.38**	**4.25**	**4.41**	**0.2499**	**0.1005**

3 HeartMuLa

In this section, we introduce **HeartMuLa** (**Heart Mu**sic **La**nguage Model), a novel framework for music generation that operates on discrete audio tokens produced by our **HeartCodec**. HeartMuLa is designed to deliver long-form, high-fidelity, and controllable music synthesis. The section is organized as follows: Sec. 3.1 presents the hierarchical architecture, which enables efficient and high-fidelity generation; Sec. 3.2 details the integration of diverse conditions for precise control. The overall architecture of HeartMuLa is illustrated in Fig. 3; Sec. 3.3 describes the training strategy and implementation details; and Sec. 3.4 presents the evaluation of the model.

3.1 Hierarchical Architecture

Following previous work Yang et al. [2023b], HeartMuLa employs a hierarchical factorization of the modeling process, initially capturing the coarse, long-range musical structure and subsequently incorporating fine-grained acoustic details. Specifically, we adopt our HeartCodec to tokenize audio into RVQ token sequence $A = [a_0, a_1, \ldots, a_{L-1}] \in [V]^{L\times K}$, where $L = Tf_a$ denotes the number of frames, and each frame $a_l = [a_{l,0}, \ldots, a_{l,K-1}]$ consists of K RVQ tokens. Let $h_{l,k}$ denote the

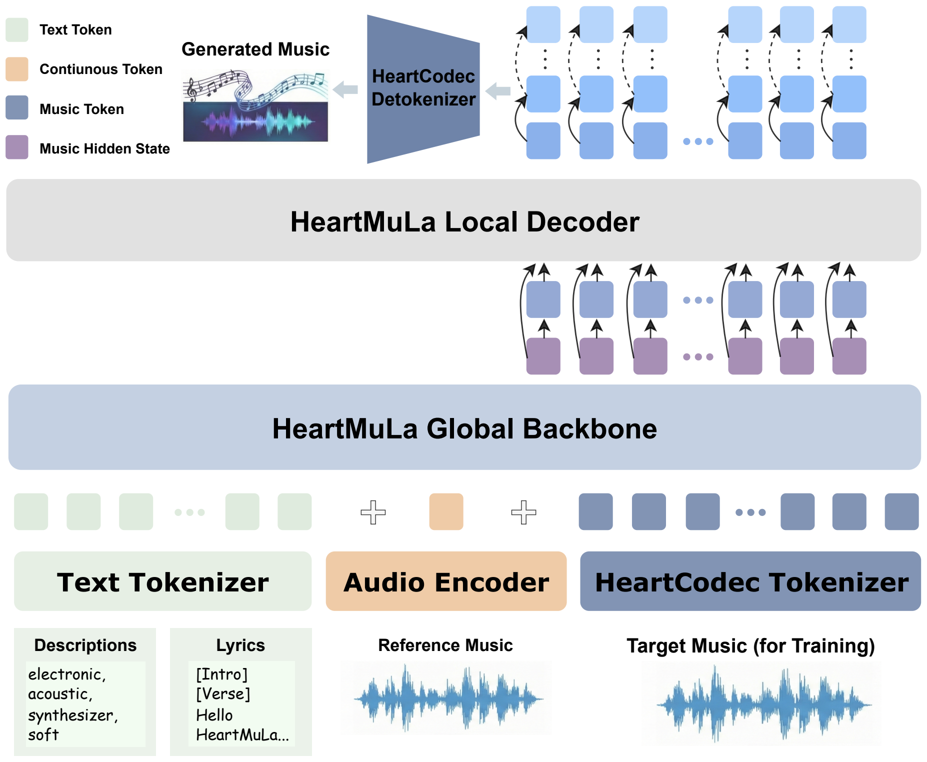

Figure 3: HeartMuLa Architecture

embedding of token $a_{l,k}$, and define $h_l = \sum_{k=0}^{K-1} h_{l,k}$ as the embedding of frame a_l. The modeling process is decomposed into global and local stages: a global transformer θ_{glo} which first models intra-frame dependencies by predicting $a_{l,0}$ at each frame l, conditioned on the preceding frame embeddings $h_{<l}$, capturing the coarse semantic information encoded in the RVQ Layer 0 code, followed by a subsequent local transformer θ_{loc} predicting the remaining tokens $a_{l,k}$ within this frame, conditioned on both the hidden state of the global transformer $\theta_{\text{glo}}(h_{<l})$ and local token embeddings $h_{l,<k}$. The overall probability is given by:

$$p(a_l \mid h_{<l}; \theta_{\text{glo}}, \theta_{\text{loc}}) = p(a_{l,0} \mid h_{<l}; \theta_{\text{glo}}) \left(\prod_{k=1}^{K-1} p(a_{l,k} \mid h_{l,<k}, \theta_{\text{glo}}(h_{<l}); \theta_{\text{loc}}) \right) \tag{7}$$

This hierarchical architecture simultaneously delivers computational efficiency and high-fidelity generation capabilities. Enhanced Efficiency is achieved through sequence factorization; the large-scale global transformer is tasked solely with predicting the base tokens of layer 0 rather than the entire multi-layer hierarchy. This strategy significantly mitigates the computational overhead and modeling complexity associated with predicting multi-stream codebooks. Furthermore, the architecture leverages the global transformer to capture coarse-grained semantic and structural patterns across frames, while offloading the synthesis of intricate acoustic details to the local transformer. These components work in tandem to elevate the overall quality of the generated audio synergistically.

3.2 Conditioning Mechanism

HeartMuLa uses lyrics, complemented by optional tags and reference audio as conditioning signals to enable precise control over the generation process.

Lyrics are annotated with structural markers such as `[intro]`, `[verse]`, and `[chorus]`, which guide the model in identifying and preserving the song's structure. These markers are retained during tokenization by the Llama-3.2 tokenizer Team [2024], yielding lyrics token sequence which is further embedded into C_{lyrics}.

Tags capture high-level musical attributes, each falling under a specific category (e.g., genre, instrument), with varying levels of influence on the music. To prioritize categories with greater impact, we empirically assign a selection probability to each category, ensuring that more influential tags, such as genre, are given higher probabilities compared to less impactful ones, such as topic. The specific categories and their selection probabilities are provided in Table 6. These selected descriptions are then encapsulated within special tokens `<tag>` and `</tag>`, followed by tokenization via the Llama-3.2 tokenizer Team [2024] and embedding to form the tag sequence, represented as C_{tag}.

Reference Audio serves as a global stylistic cue. During training, we randomly sample a 10-second segment from the ground-truth music and employ pre-trained MuQ-MuLan Zhu et al. [2025] embeddings [1] to characterize its musical style. This conditioning signal, denoted as C_{muq}, is discarded with a 50% probability during training to facilitate unconditional modeling.

Together, C_{lyrics}, C_{tag} and C_{muq} form our overall condition sequence C, which is prepended before h_0 and integrated as the prompt condition. This leads to a slight modification in the training objective compared to Eq. 7, and will be detailed in Sec. 3.3.2. Additionally, not all conditions are used during every training stage, as will be explained in Sec. 3.3.1.

Table 6: Selection Probabilities for Different Tag Categories

Category	Genre	Timbre	Gender	Mood	Instr.	Scene	Region	Topic
Prob.	0.95	0.5	0.375	0.325	0.25	0.2	0.125	0.1

3.3 Training

We adopt a four-stage progressive training paradigm, as depicted in Fig. 4, which includes warm-up, pre-training, supervised fine-tuning, and reinforcement learning. We introduce each stage briefly in Sec. 3.3.1 and corresponding training objectives in Sec. 3.3.2. Implementation details are provided in Sec. 3.3.3.

3.3.1 Four-Stage Progressive Training Paradigm

Stage 1: Warmup. In this stage, we train HeartMuLa on 30-second music segments containing lyrics. The input conditions $C = [C_{\mathrm{muq}}, C_{\text{lyrics}}]$ include the lyrics and the reference audio. The core objective of this stage is to facilitate rapid parameter convergence and establish a preliminary mastery of local acoustic texture modeling laws through dense training on short contexts, laying a foundational acoustic capability for subsequent long-sequence generation.

Stage 2: Pretraining. In this stage, HeartMuLa is trained on a large-scale dataset of full songs, using all three conditions $C = [C_{\text{tag}}, C_{\mathrm{muq}}, C_{\text{lyrics}}]$. This stage aims to leverage massive data to force the model to learn long-range temporal dependencies and global musical structures under complete conditional inputs, thereby achieving precise adherence to lyrical content and effective control over the overall musical style.

Stage 3: Supervised Finetuning. In this stage, we use AudioBox Tjandra et al. [2025] and SongEval Yao et al. [2025] to filter a high-quality subset from the full data. Then we fine-tune the model using all conditions $C = [C_{\text{tag}}, C_{\mathrm{muq}}, C_{\text{lyrics}}]$. This stage aims to improve both the synthesis quality and the fine-grained structural control of the generated music via fine-tuning optimization.

Stage 4: Direct Preference Optimization. To further elevate the perceptual quality of the generated music, we use a reinforcement alignment stage utilizing Direct Preference Optimization (DPO) Rafailov et al. [2023]. By leveraging preference data constructed with metrics including tag similarity, Phonemes Error Rate (PER), and AudioBox Tjandra et al. [2025] scores, DPO enables the model to effectively distinguish and generate superior music. This stage is specifically designed to enhance multi-dimensional generation quality, resulting in significant improvements in vocal clarity, stylistic fidelity, and overall audio coherence.

[1]We do not want to release a model that can copy speaker's timbre, and we find that MuQ-MuLan does not include any speaker timbre information. Thus, we choose MuQ-Mulan to extract the style embedding.

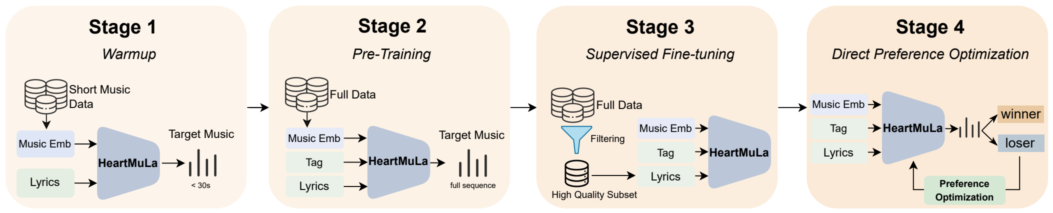

Figure 4: Four-Stage Progressive Training Paradigm

3.3.2 Optimization Objectives

Weighted CrossEntropy Loss Based on the hierarchical modeling architecture of HeartMuLa, our optimization objective is decomposed into a global loss $\mathcal{L}_0$ targeting long-term semantic modeling and residual terms $\mathcal{L}_k, (k \in [1, K-1])$ focusing on acoustic detail reconstruction. The total loss function $\mathcal{L}_{\text{total}}$ is defined as follows:

$$\mathcal{L}_{\text{total}} = \lambda_0 \mathcal{L}_0 + \frac{1}{K-1} \sum_{k=1}^{K-1} \lambda_k \mathcal{L}_k \tag{8}$$

where each term corresponds to the cross-entropy loss at the specific RVQ layer:

$$\mathcal{L}_0 = -\frac{1}{L-1} \sum_{l=1}^{L-1} \log p(a_{l,0} \mid h_{<l}, C; \theta_{\text{glo}}) \tag{9}$$

$$\mathcal{L}_k = -\frac{1}{L-1} \sum_{l=1}^{L-1} \log p(a_{l,k} \mid h_{l,<k}, \theta_{\text{glo}}(h_{<l}, C); \theta_{\text{loc}}), \forall k \in [1, K-1] \tag{10}$$

Here, λ_0 is the weight for loss at layer 0, and λ_k for the loss at the k-th residual layer. We assign more weight to layer 0 by averaging the residual terms, as layer 0 captures the coarse semantic information. The total loss function $\mathcal{L}_{\text{total}}$ integrates the prefix condition C into Eq. 7, applying the negative log-likelihood followed by weighted summation.

During the Warmup and Pretraining stages, we aim for the model to learn the acoustic feature distribution across all layers in a balanced manner. Consequently, we set the global loss coefficient to $\lambda_0 = 1.0$ and assign uniform weights to all residual layers, i.e., $\lambda_k = 1.0, \forall k \in [1, K-1]$. During the Supervised Finetuning stage, we raise the weight of global loss to $\lambda_0 = 2.0$ to emphasize musical structure and semantics, while weights for the remaining terms are set to $\lambda_k = \frac{K-k}{10}$ to attenuate their influence.

Direct Preference Optimization Loss Traditional Reinforcement Learning (RL) methods for LLMs (e.g., PPO) Schulman et al. [2017] rely on an explicit reward model and online sampling, leading to substantial computational overhead and training instability. We instead adopt Direct Preference Optimization (DPO) Rafailov et al. [2023], which casts the RL objective as supervised learning and directly optimizes the policy with preference pairs.

Given a dataset of preference pairs $\mathcal{D} = \{(C, A_{wn}, A_{ls})\}$, where C denotes the prompt (including style description, structure lyrics and optional reference music), A_{wn} represents the preferred (winning) audio sequence, and A_{ls} represents the dispreferred (losing) audio sequence, DPO derives a closed-form solution to the constrained reward maximization problem. The resulting objective bypasses explicit reward modeling by expressing the optimal policy directly in terms of the log-probability ratio between the policy model p_θ and a reference model p_{ref}:

$$\mathcal{L}_{\text{DPO}}(\theta) = -\mathbb{E}_{(C, A_{wn}, A_{ls}) \sim \mathcal{D}} \left[\log \sigma \left(\beta \cdot \Delta_\theta(C, A_{wn}, A_{ls}) \right) \right], \tag{11}$$

where $\sigma(\cdot)$ denotes the sigmoid function, β is a temperature hyperparameter controlling the deviation from the reference policy, and

$$\Delta_\theta(C, A_{wn}, A_{ls}) = \left(\log p_\theta(A_{wn}|C) - \log p_{\text{ref}}(A_{wn}|C) \right) - \left(\log p_\theta(A_{ls}|C) - \log p_{\text{ref}}(A_{ls}|C) \right). \tag{12}$$

This formulation implicitly defines the reward as $r(C, A) = \beta \log \frac{p_\theta(A|C)}{p_{\text{ref}}(A|C)}$, enabling the model to learn from preference comparisons without requiring reward labels or online generation during training.

HeartMuLa employs a hierarchical modeling process where layer 0 code is modelled by global transformer and residual codes by local transformer. This design necessitates a decomposition of the log-probability. Similar to Eq. 8, the log-probability is factorized as follows:

$$\log p_\theta(A|C) = \sum_{l=1}^{L} \log p(a_{l,0} \mid h_{<l}; \theta_{\text{glo}}) + \sum_{l=1}^{L} \sum_{k=1}^{K-1} \log p(a_{l,k} \mid h_{l,<k}, \theta_{\text{glo}}(h_{<l}); \theta_{\text{loc}}). \tag{13}$$

Substituting Eq. 13 into Eq. 11 yields the final DPO loss, which is naturally decomposed into two additive terms according to the hierarchical structure of HeartMuLa: one associated with the global semantic codes (layer-0) and another associated with the local residual acoustic codes (layers 1 to $K - 1$). Importantly, this decomposition implies that preference signals (rewards) can independently influence global semantic coherence and local acoustic details. As a result, the model can adjust its global semantic planning and local acoustic rendering in a disentangled manner, enabling more stable and targeted optimization compared to applying a single monolithic DPO objective over all tokens.

3.3.3 Implementation Details

Built upon the Llama 3.2 architecture Team [2024], HeartMuLa integrates a 3B-parameter global backbone with a 300M-parameter local decoder, optimized using the training strategy detailed in Section 3.3.

Warmup Details. We initiated training with a 10,000-hour subset randomly sampled from our 100,000-hour high-quality music corpus, segmented into 30-second clips. This stage ran on 8 NVIDIA A100 80GB GPUs for 5 epochs. We utilized the AdamW optimizer with a Cosine scheduler, setting the learning rate to $2e - 4$ and the CFG dropout probability to 0.02 to support Classifier-Free Guidance.

Pretraining Details. Subsequently, we scaled the training to the full 100,000-hour dataset on 64 NVIDIA A100 GPUs. The model was trained for another 5 epochs. While maintaining the optimizer configuration and CFG dropout, we annealed the learning rate to $2e - 5$ to ensure stability.

SFT Details. This stage employed a curated dataset of 15,000 hours of high-quality music. Training was executed on 8 NVIDIA A100 GPUs for 3 epochs, using the same learning rate $2e - 5$ and hyperparameters as the pre-training stage.

DPO Data Preparation. To construct the preference data for DPO, we employed a divergent sampling strategy. For each prompt input, the model generated four candidate audio samples with varying quality. Based on these candidates, we curated three distinct preference datasets, each prioritizing different metrics to guide the optimization process. In all cases, a preference pair (y_w, y_l) consists of a chosen winner sample and a loser sample. The specific criteria are as follows:

- **Muq-similarity-based Set:** To enhance semantic alignment, the candidate with the highest Muq-similarity score was selected as y_w, and the one with the lowest score as y_l. We enforced a margin of $sim_w - sim_l > 0.12$ to ensure sufficient discriminability, and required $sim_w > 0.3$ to guarantee the fundamental quality of the positive sample.
- **PER-based Set:** Focusing on articulation accuracy, we selected the sample with the lowest Phoneme Error Rate (PER) as y_w and the highest as y_l. A margin constraint of $|\text{PER}_w - \text{PER}_l| > 0.1$ was applied.
- **Audiobox & SongEval-based Set:** To capture holistic audio quality, y_w was identified as the sample achieving superior scores on both Audiobox Tjandra et al. [2025] and SongEval Yao et al. [2025] metrics, while y_l performed worst on both. We filtered pairs to ensure a SongEval score margin > 0.5 and an Audiobox average score margin > 0.8.

Training Configuration. For the DPO training phase, we set the learning rate to 1×10^{-7} and the KL penalty parameter β to 0.1. The model was trained for 3 epochs on a cluster of 8 NVIDIA A100 GPUs.