Efficient Autoregressive Audio Modeling via Next-Scale Prediction

Anonymous ACL submission

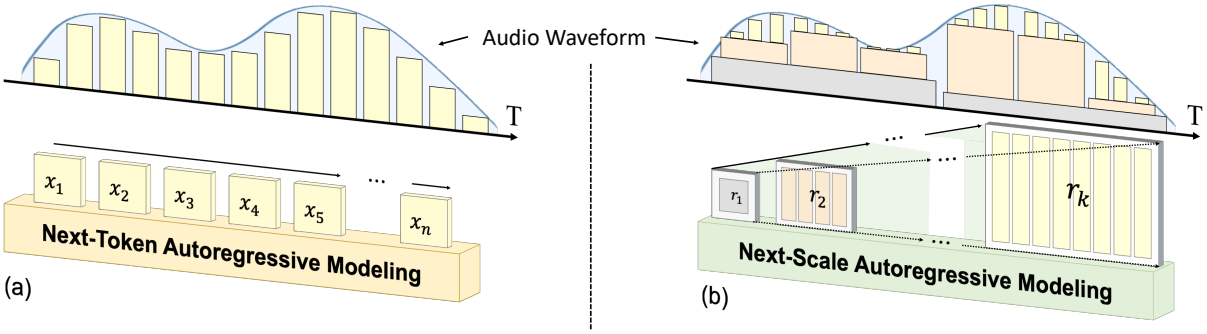


Figure 1: Autoregressive modeling of audio. (a) Next-token prediction: sequential token generation in chronological order (left to right), which aligns with the natural temporal structure of audio; (b) Next-scale prediction: multi-scale token maps are autoregressively generated from coarse to fine scales (lower to higher resolutions). Tokens are generated in parallel within each scale, which reduces about 40x the AR prediction iteration.

Abstract

Audio generation has achieved remarkable progress with the advance of sophisticated generative models, such as diffusion models (DMs) and autoregressive (AR) models. However, due to the naturally significant sequence length of audio, the efficiency of audio generation remains an essential issue to be addressed, especially for AR models that are incorporated in large language models (LLMs). In this paper, we analyze the token length of audio tokenization and propose a novel **S**cale-level **A**udio **T**okenizer (SAT), with improved residual quantization. Based on SAT, a scale-level **A**coustic **A**uto**R**egressive (AAR) modeling framework is further proposed, which shifts the next-token AR prediction to next-scale AR prediction, significantly reducing the training cost and inference time. To validate the effectiveness of the proposed approach, we comprehensively analyze design choices and demonstrate the proposed AAR framework achieves a remarkable **35**$\times$ faster inference speed and **+1.33** Fréchet Audio Distance (FAD) against baselines on the AudioSet benchmark.

1 Introduction

Autoregressive (AR) modeling (Achiam et al., 2023; Sun et al., 2024) has been widely used in the generation domain, which typically involves two steps - token quantization (Esser et al., 2021; Yu et al., 2021) and next-token prediction (Achiam et al., 2023; Touvron et al., 2023). Specifically, the token quantization aims to convert the inputs to a sequence of discrete tokens and the next-token prediction models the conditional distribution of one token based on previous ones. AR approaches have shown significant success in textual modeling, e.g., large language models (LLMs) (Vaswani et al., 2017; Devlin et al., 2018; Touvron et al., 2023; Achiam et al., 2023) and even visual modeling (Dosovitskiy et al., 2020; Chang et al., 2022). However, despite its effectiveness, AR-based audio generation remains under-explored.

Unlike natural language which is discrete and can be easily tokenized into a short series of tokens, audio demonstrates more challenges to be discretized without losing perceptual quality given its long sequence and continuity nature. Previous approaches (Défossez et al., 2022; Yang et al., 2023; Kumar et al., 2024; Zeghidour et al., 2021) leverage multi-stage residual quantization (RQ) (Lee et al., 2022) to model the raw waveform with different frequencies. However, the multi-stage RQ will significantly increase the token length, leading to difficulty in the subsequent next-token prediction. Another paradigm (Baevski et al., 2020) focuses on the semantics of the waveform and leverages pre-trained models (e.g., Hubert (Hsu et al., 2021)) to cluster the embeddings in the semantic

space and then quantize the embeddings based on cluster centers. Though semantic embeddings can successfully reconstruct the waveform, the reconstruction quality and generalization capability are bottlenecked by the pre-trained encoder.

In addition, compared to text and images, audio waveform typically has a much longer sequence length due to the high sampling rates, such that about 960000 sequence length in 1 min audio clip with 16kHz. Since AR models predict tokens in a sequential manner, the inference cost is quadratically correlated to the sequence length, making the AR-based audio generation slow and computationally expensive, as illustrated in Fig. 1 (a).

In this paper, we explore the **S**cale-level **A**udio **T**okenizer and Multi-Scale **A**coustic **A**uto**R**egressive Modeling in audio generation, as illustrated in Fig. 1 (b). On the one hand, to shorten the audio token length, we utilize a scale-level audio tokenizer (SAT) which improves the traditional residual quantization with a multi-scale design and compresses the token length according to the scale index. On the other hand, we further shorten the inference step during the autoregressive prediction. Based on the multi-scale audio tokenizer, we propose acoustic autoregressive modeling (AAR) which models the audio tokens with a next-scale paradigm. Since each scale contains multiple audio tokens, the AAR can lead to much fewer autoregressive step numbers during inference compared to the traditional token-level modeling. By reducing both the token length and the autoregressive step number, our approach achieves not only a superior generated audio quality but also a remarkably faster (about $35\times$) inference speed.

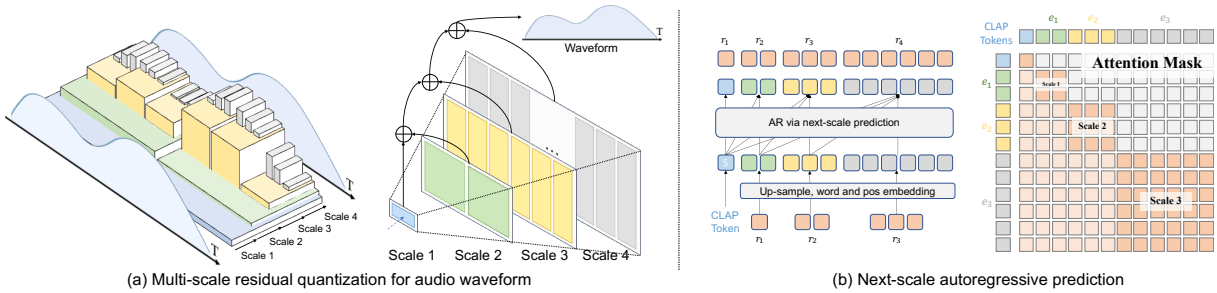
Our contribution can be summarized in threefold:

- We present **S**cale-level **A**udio **T**okenizer (SAT) for audio reconstruction which can efficiently compress audio sequence to tokenizers with different scales.
- Based on SAT, we introduce scale-level **A**coustic **A**uto**R**egressive modeling (AAR), significantly reducing the inference latency and training cost.
- Extensive experiments are conducted to analyze the performance of the proposed approach, providing insights into its capabilities and potential applications in the field of audio synthesis.

2 Related Works

Raw audio discretization. Before the development of Variational Autoencoders (VAEs) (Van Den Oord et al., 2017; Razavi et al., 2019), converting continuous domains into discrete representations was a significant challenge in generative modeling. VAEs facilitate the effective quantization of inputs into structured priors using powerful encoder-decoder networks, allowing manipulation in tasks like generation and understanding (Achiam et al., 2023; Touvron et al., 2023; Caillon and Esling, 2021). Recent innovations, such as VQ-GAN (Esser et al., 2021) and RQGAN (Lee et al., 2022), have further advanced these priors, improving model generalization and inspiring numerous works in audio discretization (Oord et al., 2016; Caillon and Esling, 2021). In the audio domain, Encodec (Défossez et al., 2022) employs an architecture similar to SoundStream (Zeghidour et al., 2021), using an encoder-decoder model to reconstruct audio, incorporating residual quantization and a spectrogram-style discriminator to enhance audio quality. In contrast, HIFI-codec (Yang et al., 2023) uses group residual quantization to refine the representation in the initial quantization layer. Kumar et al. (Kumar et al., 2024) have made significant contributions to audio reconstruction by introducing multi-spectrogram loss and quantizer dropout to enhance bitrate efficiency. Building upon these advances, our work poses an important question: can we use fewer tokens to represent low-frequency information, thereby efficiently reducing the computational burden while maintaining high-quality audio reconstruction? To address this, we propose a **S**cale-level **A**udio **T**okenizer, which encodes audio on different scales, capturing hierarchical features that improve both the efficiency and quality of audio generation and reconstruction.

Diffusion-based audio generation Diffusion models (Yoon et al., 2023; Song et al., 2020; Lu et al., 2022), initially introduced by Sohl-Dickstein et al., have made significant strides in recent years due to their ability to generate high-quality audio by progressively transforming noise into coherent signals. These models have been widely adopted in various audio applications, including speech synthesis, music generation (Hawthorne et al., 2022), and general audio synthesis (Kong et al., 2020b), because of their robustness and flexibility in handling the complexities of audio data. This has inspired researchers to extend diffusion models to dif-



(a) Multi-scale residual quantization for audio waveform (b) Next-scale autoregressive prediction

Figure 2: Our model involves two distinct training phases. **Stage 1**: Scale-level Audio Tokenizer (SAT) to encode an audio sample into a series of K tokens scales, donated as $\mathcal{R} = (r_1, r_2, \ldots, r_K)$. Each scale encodes information in different frequencies of the audio waveform. **Stage 2**: Acoustic AutoRegressive (AAR) modeling via next-scale prediction relies on the pre-trained SAT to predict each scale-level token r_i by conditioning on all previously predicted scales $r_{<i}$ and a CLAP token (Wu et al., 2023) as the start token. The CLAP token is derived from ground truth audio. During training, we use the standard cross-entropy loss and the attention mask as figured above to ensure that each r_i can only be attributed by $r_{\leq i}$ and the start token.

ferent modalities, consistently demonstrating their adaptability and strength in creating realistic and detailed synthetic outputs. However, the diffusion process inherently presents several challenges for audio generation: (1) high computational costs and significant inference times due to the iterative nature of the diffusion process, and (2) difficulty integrating with mainstream intelligent systems due to differences in representation. These challenges motivate us to explore audio generation using large language models (LLMs) (Vaswani et al., 2017; Devlin et al., 2018; Touvron et al., 2023; Achiam et al., 2023), which offer an alternative approach with potentially more efficient computational demands and compatibility with existing intelligent systems.

Autoregressive modeling The autoregressive model (Chowdhery et al., 2023; Hoffmann et al., 2022), as a different approach from diffusion models, leverages efficient Large Language Models (LLMs) (Vaswani et al., 2017; Devlin et al., 2018; Touvron et al., 2023; Achiam et al., 2023) to generate the next tokens sequentially to construct the output. Due to its sequential nature, autoregressive models have excelled in text generation, machine translation, and other sequence prediction tasks. Recently, autoregressive models have also made significant processes in the image generation domain (Chang et al., 2022; Sun et al., 2024). By treating image pixels or patches as sequences, these models can generate high-quality images by sequentially predicting each part of the image. However, the application of autoregressive models to raw audio generation remains challenging. The primary limitation is the sheer number of tokens required to represent raw audio data. Audio signals have a high temporal resolution, meaning that accurately capturing the nuances of sound requires a large number of tokens. This results in increased computational complexity and longer generation times, making it difficult to achieve real-time audio generation with current autoregressive models. To mitigate these limitations, our paper proposes a novel direction on employing **S**cale-level **A**udio **T**okenizer to encode raw audio at different scales and generate it using **A**coustic **A**uto**R**egressive modeling via next-scale prediction, thereby enhancing audio generation efficiency and quality.

3 Method

Our approach consists of two major stages: (1) In the first stage, we train a **S**cale-level **A**udio **T**okenizer (SAT) to convert continuous audio signals into discrete tokens using multi-scale residual quantization. (2) The second stage reformulates the **A**coustic **A**uto**R**egressive modeling (AAR) in a next-scale manner and models the tokens obtained from the frozen SAT tokenizer with a transformer structure.

3.1 Baseline Audio Tokenizer

Audio quantization. Consider an audio signal $a \in \mathbb{R}^{C \times T}$, where C represents the number of audio channels and T is the number of samples over the duration of the signal. Traditional approach (Kumar et al., 2024; Défossez et al., 2022; Yang et al., 2023) in audio tokenizer often involves a 1D convolutional-based autoencoder frameworks to compress audio waveform to latent space $x \in \mathbb{R}^{l \times d}$

where l is the token length and then utilizes a vector quantization to quantize the latent tokens:

$$x = \mathcal{E}(a), \quad \hat{x} = \mathcal{Q}(x), \quad \hat{a} = \mathcal{D}(\hat{x}) \tag{1}$$

where $\mathcal{E}(\cdot)$ donates encoder, $\mathcal{Q}(\cdot)$ a vector quantizatier, and $\mathcal{D}(\cdot)$ a decoder. A vector quantizer $\mathcal{Q}$ maps each feature vector in the latent space x to the closest vector in a learnable codebook $Z \in \mathbb{R}^{d \times V}$ with V vectors of dimension d. Specifically, vector quantization $\hat{x} = \mathcal{Q}(x)$ involves looking up the closest match for each feature vector in x with vectors in Z by minimizing Euclidean distance:

$$\hat{x} = \text{argmin}_{z \in Z} ||x - z||_2 \tag{2}$$

where $\hat{x}$ represents the quantized output and x is the input to the quantizer.

However, due to the complexity of the audio waveform, particularly in handling frequency-specific information, a residual quantization approach is typically employed. In residual quantization, a sequence of vector quantizers $\mathcal{Q} = \{\mathcal{Q}_1, \mathcal{Q}_2, \cdots, \mathcal{Q}_K\}$ is used, where each quantizer $\mathcal{Q}_i$ iteratively quantizes the residual error from the previous step. Specifically, after each quantization step, the residual error is computed as $\delta_i = x_i - \hat{x}_i$ and passed to the next quantizer as the input $x_i = \delta_{i-1}$. The final quantized representation $\hat{f}$ is obtained by summing the outputs from all quantizers

$$\hat{x} = \sum_{i=1}^{r} \hat{x}_i \tag{3}$$

which is then decoded by the decoder $\mathcal{D}(\hat{x})$ to produce the reconstructed output $\hat{a}$.

Loss function. To train audio quantized autoencoder, we leverage a combination of loss functions including the reconstructed time-domain loss $\mathcal{L}_t$, reconstructed frequency domain loss $\mathcal{L}_f$, discriminative loss $\mathcal{L}_G$, residual quantization loss $\mathcal{L}_{vq}$, and commitment loss $\mathcal{L}_{commit}$ (Défossez et al., 2022) :

$$\mathcal{L} = \lambda_t \mathcal{L}_t + \lambda_f \mathcal{L}_f + \lambda_G \mathcal{L}_G + \mathcal{L}_{vq} + \lambda_{com} \mathcal{L}_{com}. \tag{4}$$

Specifically, reconstructed time-domain loss measures the absolute difference between a and $\hat{a}$ as

$$\mathcal{L}_t = |a - \hat{a}|| \tag{5}$$

and frequency domain loss assesses the difference over mel-spectrograms across n time scales as

$$\mathcal{L}_f = \sum_{i=1}^{n} ||\mathcal{S}_i(a) - \mathcal{S}_i(\hat{a})|| + ||\mathcal{S}_i(a) - \mathcal{S}_i(\hat{a})||_2^2 \tag{6}$$

where $\mathcal{S}_i$ represents the transformation to the mel-spectrogram at scale i. The discriminative loss is derived from a multi-scale STFT discriminator, as introduced in (Zeghidour et al., 2021) to ensure the model captures high-fidelity audio features across various time-frequency scales. The vector quantization loss encourages the encoded features to match the codebook vectors, and the commitment loss penalizes deviations from these vectors, ensuring that the encoder commits to the quantized space as

$$\begin{aligned} \mathcal{L}_{vq} &= \sum_{i=1}^{r} ||\text{sg}(x_i) - z_i||_2^2 \\ \mathcal{L}_{com} &= \sum_{i=1}^{r} ||x_i - \text{sg}(z_i)||_2^2. \end{aligned} \tag{7}$$

Analysis. The baseline audio tokenizer can successfully discretize audio tokens. However, due to the residual quantization, the token length representing each audio will be significant which severely hinders the efficiency in the autoregressive modeling. Considering each quantizer in residual quantization basically divides and represents the audio into different frequency bands (Défossez et al., 2022; Kumar et al., 2024; Yang et al., 2023), we aim to further adjust the token length based on its represented frequencies, i.e., lower-frequency parts can be represented with fewer tokens. To this end, we introduce the **S**cale-level **A**udio **T**okenizer to reduce the number of tokens being used.

3.2 Scale-level Audio Tokenizer

In Scale-level Audio Tokenizer (SAT), we employ the same encoder-decoder architecture as baseline tokenizer (Défossez et al., 2022) but incorporate multi-scale residual quantization (MSRQ) to enhance efficiency and flexibility in audio representation. In MSRQ, as shown in Fig. 2 (a), the quantizer $\mathcal{Q}_i$ is defined the same as the baseline setting as $r_i = \mathcal{Q}_i(r_{i-1})$ while the feature map r_{i-1} is first downsampled from its original dimension $l_K \times d$ to a lower resolution $l_k \times d$ where K is the scale number of the last index and k is the scale number of the correct index. After downsampling, the look-up procedure is performed to match each feature vector with the closest codebook vector Z_i. After the look-up, the processed quantized vector z_i is upsampled back to the original dimension $l_K \times d$ to ensure consistency across scales. Due to the loss of high-frequency information from downsampling, we employ a 1D convolutional layer after upsampling to restore the lost details and enhance

the fidelity of the reconstructed audio. Specifically, this convolutional layer processes the upsampled feature vectors according to the equation

$$\phi(\hat{r}) = \gamma \times \text{conv}(\hat{r}) + (1 - \gamma) \times \hat{r} \tag{8}$$

where $\text{conv}(\cdot)$ applies a 1D convolution with a kernel size of 9. This design effectively combines the original features with the transformed outputs, while preserving the reparameterization inherent to vector quantization, controlled by the quantization residual ratio γ. In the Appendix, we provide a pseudo-code for the scale-level audio tokenizer.

3.3 Acoustic AutoRegressive Modeling

Vanilla autoregressive modeling. Autoregressive modeling is first introduced by (Sutskever et al., 2014; Bahdanau et al., 2014) and quickly spread to different modalities such as image (Sun et al., 2024), video (Weissenborn et al., 2019) and 3D modeling (Siddiqui et al., 2024). In autoregressive modeling, a sequence of data points is modeled as a product of conditional probabilities. For a sequence $x = (x_1, x_2, \ldots, x_T)$, its joint distribution can be expressed and modeled as

$$p(x_1, x_2, ..., x_T) = \prod_{t=1}^{T} p(x_t | x_1, x_2, ...x_{t-1}). \tag{9}$$

This approach is widely used across various domains due to its flexibility and ability to capture dependencies within data. For any continuous modality, it is traditional to first train a tokenizer to discretize the input into tokens, which can then be modeled using a discrete categorical distribution. This step involves mapping the continuous data to a sequence of discrete tokens $x = (x_1, x_2, \ldots, x_T)$ that are fed into an autoregressive model to predict the next token in the sequence, based on the preceding tokens. In the context of transformers, which have become the dominant architecture for autoregressive modeling, the attention mechanism plays a crucial role in training. The attention mechanism allows the model to focus on different parts of the input sequence when making predictions. To ensure that the model adheres to the autoregressive property, where each token x_i is predicted based only on previous tokens $x_1, x_2, \ldots, x_{i-1}$, an attention mask is applied. Mathematical, the attention mask M is defined as

$$M_{ij} = \begin{cases} 1, & \text{if } i \leq j \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

where guarantee the modeling's performance on predicting x_i is only relevant to its preceding tokens.

After the completion of training of such a model P using cross-entropy loss, it can efficiently handle complex dependencies and generate new samples by sequentially predicting each token conditioned on its predecessors (Achiam et al., 2023; Touvron et al., 2023; Sun et al., 2024).

This capability makes autoregressive models well-suited for generating data that requires a coherent and consistent sequence. However, their capacity for audio generation is still under-explored due to the huge sequence length required for audio data. The sheer number of tokens needed to represent even short audio clips can lead to computational inefficiencies and challenges in maintaining temporal coherence. To efficiently solve such a challenge, we combine the unique property of our SAT to efficiently generate audio via scale-level **A**coustic **A**uto**R**egressive modeling.

Acoustic autoregressive modeling. To shorten the inference step, we propose Acoustic autoregressive modeling (AAR). This approach, distinct from traditional vanilla autoregressive models that predict token sequences one by one, involves predicting across different scales. Attributed by SAT, our method represents an audio sample as a series of scale-level representations:

$$\mathcal{R} = (r_1, r_2, \ldots, r_K) \tag{11}$$

By efficiently expressing it as joint modeling, the audio sequence is defined as:

$$p(\mathcal{R}) = \prod_{i=1}^{K} p(r_i | r_1, r_2, ..., r_{i-1}) \tag{12}$$

In this formulation, each r_i represents a distinct scale in the hierarchical representation of the audio signal. The model predicts each scale by conditioning on all previously predicted scales, effectively capturing both global structures and fine-grained details of the audio. This hierarchical approach reduces the complexity associated with long sequence lengths by leveraging multi-scale dependencies, thereby enhancing the model's efficiency and ability to maintain temporal coherence. To successfully implement our method, we modify the attention mask M for each scale r_i to focus only on the relevant scales:

$$M_{ij} = \begin{cases} 1, & \text{if } i \leq j \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

ID	Method	# quantizers	# tokens	FAD↓	MEL↓	STFT↓
1	Encodec (Défossez et al., 2022)	10	750	1.39	1.33	1.97
2	Our SAT	16	455	$1.09_{-0.30}$	$1.33_{-0.00}$	$1.98_{+0.01}$

Table 1: Performance of Encodec v.s. Our SAT. We evaluated FAD, MEL Distance, and STFT Distance on the Audioset validation set. ↓ means lower is better.

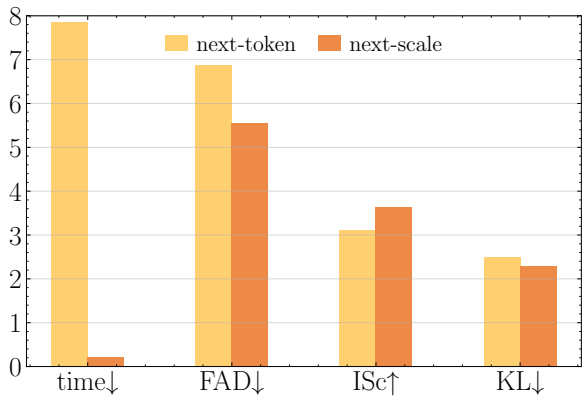


Figure 3: Performance of autoregressive model when classifier-free guidance is 10. next-token: AR via next-token prediction; next-scale: our AAR.

This attention mask ensures that the model only attends to $r_1, r_2, \ldots r_{i-1}$ when predicting r_i ignoring future scales and reducing unnecessary computations. A detailed description of the implementation is summarized in the Appendix.

4 Experiment

4.1 Evaluation Metrics and Settings

We evaluate FAD (Kilgour et al., 2018), MEL distance (Kumar et al., 2024), and STFT distance (Kumar et al., 2024) as reference for reconstruction, and FAD (Kilgour et al., 2018), ISc and KL (Salimans et al., 2016) for generation. FAD, built upon VGGish (Chen et al., 2020), is the metric to indicate the similarity of the generated and target samples effectively. MEL distance quantifies the difference in mel-spectrogram features, and STFT distance measures the short-time Fourier transform discrepancies between the generated and target audio signals, which focus more on high-frequency information for audio. Additionally, ISc, simulating its performance on image generation, is used to evaluate the generated sample diversity and quality. KL divergence is utilized to measure the difference between the probability distributions of the generated and target samples.

We conducted all experiments on the AudioSet (Gemmeke et al., 2017) dataset. To effectively evaluate the performance of our audio tokenizer, we divided the original 10-second evaluation set into n segments, each matching the window size of our model for reconstruction. After reconstructing these segments, we reassembled them into a complete audio stream. For autoregressive generation, we randomly selected one segment from the evaluation set and used it as the ground truth.

4.2 Implementation Details

Tokenizer. In stage 1, we utilize multi-scale residual quantization (MSRQ) of codebook size 1024 with the soundstream autoencoder framework (Zeghidour et al., 2021). The model is trained for 100 epochs using the Adam optimizer with $\beta_1 = 0.5$ and $\beta_2 = 0.9$. We apply a cosine learning rate scheduler with initial learning rate 3e-4 and setting the loss weights to $\lambda_t = 0.1, \lambda_f = 3, \lambda_G = 3$, and $\lambda_{com} = 1$. Our discriminator updated 2/3 times during training.

Transformer. In stage 2, our primary focus is on scale-level acoustic autoregressive modeling. To achieve this, we employ a GPT-2-style transformer (Radford et al., 2019) with adaptive normalization (Zhang et al., 2018) and depth of 16. We utilize CLAP audio embeddings (Wu et al., 2023) as our start tokens. Since one-second audio segments often contain limited meaningful information, we opt to use 10-second audio embeddings to capture richer context, even when generating one-second clips. For training, we adopt the AdamW optimizer with a learning rate of 1e-4, using a linear learning rate scheduler. Additionally, we apply a weight decay of 0.05 and implement warmup settings with an initial warmup proportion of 0.005 and an end warmup proportion of 0.01.

4.3 Main Results Analysis

For tokenization, as illustrated in Tab 1, our proposed SAT tokenizer suppresses the baseline Encodec (Défossez et al., 2022) by 0.3 FAD, despite using fewer tokens (750 tokens v.s. 455 tokens). This demonstrates that by increasing quantization while reducing the number of tokens, we can efficiently improve reconstruction quality while using

# scales	# tokens	FAD	MEL	STFT
10	207	1.81	1.55	2.08
16	303	1.52	1.46	1.80

Table 2: Ablation study of SAT performance in number of scales.

Scheduler	# tokens	FAD	MEL	STFT
Logarithmic	303	1.52	1.46	1.80
Quadratic	455	1.40	1.37	1.86
Linear	601	1.38	1.39	2.02

Table 3: Ablation study on scale setting of SAT. All scale settings are trained in the same numbers of scale.

Window	FAD	MEL	STFT
1s	1.22	1.36	1.85
5s	1.29	1.41	1.93

Table 5: Ablation study on temporal window.

STFTD	MPD	MSD	FAD	MEL	STFT
✓	✗	✗	1.38	1.36	1.76
✓	✓	✓	2.29	1.65	2.12

Table 6: Ablation study on discriminator choice. STFTD stands for Multi-scale short-time fourier transform discriminator, MPD stands for multi-period discriminator, MSD stands for multi-scale discriminator.

fewer tokens.

For the audio generation, we introduce an autoregressive model with next-token prediction as the baseline. To ensure a fair comparison, we employ two encoders (Encodec (Défossez et al., 2022) for AR and our SAT for AAR) with similar performance. We find that our proposed AAR shows superior performance in terms of both latency and audio quality. As shown in Fig. 3, the next-scale prediction demonstrates a remarkable improvement in audio generation, achieving a 35x speed improvement (0.225s v.s. 7.866s) and generation enhancement (FAD 5.55 v.s. 6.88). More analysis of training costs is available in the Appendix.

4.4 Ablation Experiments

We conduct ablation experiments to validate the effectiveness of the components in SAT and AAR.

Effect of discriminator We explored multiple discriminator configurations to optimize the performance of our Scale-level Audio Tokenizer (SAT). As illustrated in Tab 6, we tested two different discriminator setups: one using only a multi-scale short-time Fourier transform (STFT) discriminator (Zeghidour et al., 2021) and another combining the multi-scale STFT discriminator with a multi-period discriminator (MPD) (Kong et al., 2020a) and a multi-scale discriminator (MSD) (Kumar et al., 2019). Our results indicate that using only a multi-scale STFT discriminator is sufficient for effective reconstruction.

Latent dim.	FAD	MEL	STFT
8	1.47	1.55	2.15
16	1.38	1.52	2.14
32	1.60	1.43	2.05
64	1.09	1.33	1.98

Table 4: Ablation study on latent dimension. We fix the scale to 16 and use the same quadratic scale setting. "Latent dim." represents dimension of latent representation.

Effect of the scale setting. To find the optimal combination of SAT configuration, we start with Encodec in 128 latent dimensions with 10 quantizers (Défossez et al., 2022) and test multiple scales with shared codebooks of different sizes and individual codebooks for each scale. In particular, Tab 2 shows that enlarging the scale to 16 consistently improved audio quality. As illustrated in Tab 3, we tested the performance of linear, quadratic and logarithmic scheduling on 16 scales: linear scheduling provides a balanced number of tokens for each scale; quadratic scheduling focuses more on the early or late stages of the process; and logarithmic scheduling offers a more gradual progression. We believe the suboptimal performance observed in logarithmic scheduling is due to its lack of high-frequency information representation at larger scales even though it also builds a complete information flow for audio. Quadratic scheduling, in particular, proved to be more efficient, requiring fewer tokens than linear scheduling (455 v.s. 601) and also achieves comparable reconstruction performance in audio quality.

To further improve the model's capacity, we fixed the decoder dimension to 1024 and tested latent dimensions of 8, 16, 32, and 64. As Tab 4 indicated, our SAT achieves its superior performance in the latent dimension of 64.

Effect of temporal windows change. To effectively validate the performance of our scale scheduling in audio reconstruction, we train our SAT using

ID	Description	FAD↓	IS↑	KL↓	Latency↓ (s)
1	Vanila AR	10.05	2.42	3.01	7.86
2	AAR	$9.24_{-0.81}$	$2.69_{+0.27}$	$2.94_{-0.07}$	$0.21_{-7.21}$
3	+ Attn. Norm	$8.80_{-1.25}$	$2.80_{+0.38}$	$2.79_{-0.22}$	$0.25_{-7.61}$
4	+ CFG	$6.44_{-3.61}$	$3.52_{+0.90}$	$2.32_{-0.69}$	$0.25_{-7.61}$
5	+ Top-k	$6.25_{-3.81}$	$3.59_{+1.17}$	$2.30_{-0.71}$	$0.25_{-7.61}$
6	+ Top-p	$6.01_{-4.04}$	$3.68_{+1.26}$	$2.27_{-0.74}$	$0.25_{-7.61}$

Table 7: Ablation study on components of AAR. vanilla AR and AAR are implemented in GPT-2 style transformer with adaptive layer normalization; "Attn. Norm" represents normalizing q and k into unit vector before attention; "CFG" means classifier free guidance scale of 2; Top-k and Top-p are sampling strategies where Top-k randomly selects from the top 200 indices, and Top-p (nucleus sampling) selects tokens with a cumulative probability of 0.95.

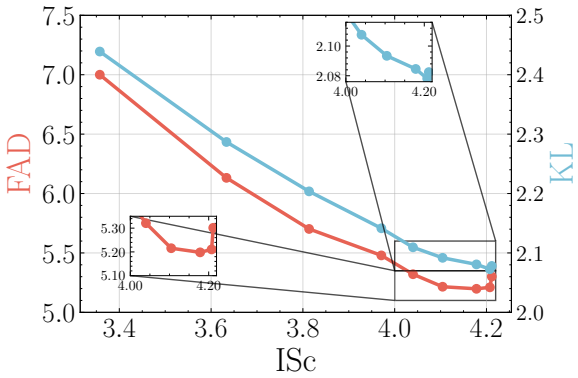


Figure 4: Performance of AAR in different classifier-free guidance scales from 2 to 18 (left to right), with each point incremented by 2. The red line represents Fréchet Audio Distance (FAD) v.s. Inception Score (ISc), while the blue line represents Kullback-Leibler divergence (KL) vs. Inception Score (ISc).

5-second audio windows with the $5\times$ original quantizer setting. This approach allows us to assess our SAT's ability to handle varying temporal dimensions and capture essential audio features over different time scales. By experimenting with different window sizes, we aim to determine the optimal configuration for maintaining high reconstruction quality while maximizing efficiency. The results of these experiments are presented in Tab 5. we find that the reconstruction quality between 1-second and 5-second windows is similar, suggesting that our SAT performs well across diverse time windows, maintaining consistent quality and demonstrating robustness in handling varying temporal scales.

Effect of AAR and sampling technique. We evaluate our AAR with the same setting as the baseline vanilla AR with roadmap shown in Tab 7. We notice that our AAR can not only improve the generation abilities (FAD, IS and KL), but also significantly reduce the inference time to an acceptable range. Moreover, the introduction of attention normalization can stabilize the training and further enhance the model's performance, leading to improved FAD and IS scores. The addition of classifier-free guidance (CFG) and advanced sampling techniques such as top-k and top-p sampling continues to push the boundaries of audio generation quality.

Effect of classifier-free guidance. As shown in Fig. 4, we evaluates the relationship between the Inception Score (ISc) with Fréchet Audio Distance (FAD) and Inception Score (ISc) with KL divergence (KL) across different Classifier-Free Guidance (CFG) scales (Ho and Salimans, 2022). We find that as the CFG scales increases, the ISc improves, while both FAD and KL metrics converge and stabilize at CFG = 14 and finally achieving FAD 5.19.

5 Conclusion

In this paper, we introduced a novel approach for audio generation using a multi-scale autoregressive model via next-scale prediction. The proposed framework leverages the scale-level audio tokenizer, which efficiently compresses audio sequences into tokenizers of varying scales, thereby improving efficiency while maintaining high fidelity. Through comprehensive experiments, we demonstrated the superior performance of our method in generating high-quality audio compared to traditional autoregressive methods.

Our approach provides an efficient solution for audio generation. By incorporating a multi-scale residual quantization technique, the model effectively reduces the sequence length required for generation, leading to enhanced efficiency and reduced computational demands.

6 Limitations.

In spite of the high performance of next-scale prediction in general audio generation with text control, there are limitations that need to be addressed in future work. These include further exploration of the model's capabilities in diverse audio environments and its integration with speech processing systems.

References

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

Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. 2020. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in neural information processing systems*, 33:12449–12460.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Antoine Caillon and Philippe Esling. 2021. Rave: A variational autoencoder for fast and high-quality neural audio synthesis. *arXiv preprint arXiv:2111.05011*.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. 2022. Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11315–11325.

Honglie Chen, Weidi Xie, Andrea Vedaldi, and Andrew Zisserman. 2020. Vggsound: A large-scale audio-visual dataset. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 721–725. IEEE.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Alexandre Défossez, Jade Copet, Gabriel Synnaeve, and Yossi Adi. 2022. High fidelity neural audio compression. *arXiv preprint arXiv:2210.13438*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Patrick Esser, Robin Rombach, and Bjorn Ommer. 2021. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12873–12883.

Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. 2017. Audio set: An ontology and human-labeled dataset for audio events. In *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 776–780. IEEE.

Curtis Hawthorne, Ian Simon, Adam Roberts, Neil Zeghidour, Josh Gardner, Ethan Manilow, and Jesse Engel. 2022. Multi-instrument music synthesis with spectrogram diffusion. *arXiv preprint arXiv:2206.05408*.

Jonathan Ho and Tim Salimans. 2022. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. 2021. Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM transactions on audio, speech, and language processing*, 29:3451–3460.

Kevin Kilgour, Mauricio Zuluaga, Dominik Roblek, and Matthew Sharifi. 2018. Fr\'echet audio distance: A metric for evaluating music enhancement algorithms. *arXiv preprint arXiv:1812.08466*.

Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. 2020a. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. *Advances in neural information processing systems*, 33:17022–17033.

Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. 2020b. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*.

Kundan Kumar, Rithesh Kumar, Thibault De Boissiere, Lucas Gestin, Wei Zhen Teoh, Jose Sotelo, Alexandre De Brebisson, Yoshua Bengio, and Aaron C

Courville. 2019. Melgan: Generative adversarial networks for conditional waveform synthesis. *Advances in neural information processing systems*, 32.

Rithesh Kumar, Prem Seetharaman, Alejandro Luebs, Ishaan Kumar, and Kundan Kumar. 2024. High-fidelity audio compression with improved rvqgan. *Advances in Neural Information Processing Systems*, 36.

Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. 2022. Autoregressive image generation using residual quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11523–11532.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. 2022. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in Neural Information Processing Systems*, 35:5775–5787.

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. 2019. Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems*, 32.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved techniques for training gans. *Advances in neural information processing systems*, 29.

Yawar Siddiqui, Antonio Alliegro, Alexey Artemov, Tatiana Tommasi, Daniele Sirigatti, Vladislav Rosov, Angela Dai, and Matthias Nießner. 2024. Meshgpt: Generating triangle meshes with decoder-only transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19615–19625.

Jiaming Song, Chenlin Meng, and Stefano Ermon. 2020. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*.

Peize Sun, Yi Jiang, Shoufa Chen, Shilong Zhang, Bingyue Peng, Ping Luo, and Zehuan Yuan. 2024. Autoregressive model beats diffusion: Llama for scalable image generation. *arXiv preprint arXiv:2406.06525*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Aaron Van Den Oord, Oriol Vinyals, et al. 2017. Neural discrete representation learning. *Advances in neural information processing systems*, 30.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Dirk Weissenborn, Oscar Täckström, and Jakob Uszkoreit. 2019. Scaling autoregressive video models. *arXiv preprint arXiv:1906.02634*.

Yusong Wu, Ke Chen, Tianyu Zhang, Yuchen Hui, Taylor Berg-Kirkpatrick, and Shlomo Dubnov. 2023. Large-scale contrastive language-audio pretraining with feature fusion and keyword-to-caption augmentation. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE.

Dongchao Yang, Songxiang Liu, Rongjie Huang, Jinchuan Tian, Chao Weng, and Yuexian Zou. 2023. Hifi-codec: Group-residual vector quantization for high fidelity audio codec. *arXiv preprint arXiv:2305.02765*.

TaeHo Yoon, Joo Young Choi, Sehyun Kwon, and Ernest K Ryu. 2023. Diffusion probabilistic models generalize when they fail to memorize. In *ICML 2023 Workshop on Structured Probabilistic Inference Generative Modeling*.

Jiahui Yu, Xin Li, Jing Yu Koh, Han Zhang, Ruoming Pang, James Qin, Alexander Ku, Yuanzhong Xu, Jason Baldridge, and Yonghui Wu. 2021. Vector-quantized image modeling with improved vqgan. *arXiv preprint arXiv:2110.04627*.

Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. 2021. Soundstream: An end-to-end neural audio codec. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:495–507.

Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. 2018. Self-attention generative adversarial networks. In *International Conference on Machine Learning*, pages 7354–7363.

Algorithm 1 Multi-scale residual quantization

Input: Raw Audio Signal $\mathcal{A}$
Parameter: Encoder $\mathcal{E}$, Decoder $\mathcal{D}$, Quantizer $\mathcal{Q}_{i=1}^{K}$
Hyper-parameter: multi-scale Resolution $(t_k)_{k=1}^{K}$, interpolation method ϕ

$f = \mathcal{E}(\mathcal{A})$
$R = []$
for k in $(1, 2, ...K)$ **do**
  if $k = K$ **then**
    $r_k = \mathcal{Q}_k(f)$
    $z_k = \mathit{lookup}(Q_k, r_k)$
  else
    $r_k = \mathcal{Q}_k(\mathit{interpolate}(f, t_k))$
    $z_k = \mathit{lookup}(Q_k, r_k)$
    $z_k = \mathit{interpolate}(z_k, t_K)$
  end if
  $R = R + r_k$
  $f = f - \phi(z_k)$
end for
return R

Algorithm 2 Multi-scale AR Generation

Input: Text $\mathcal{T}$
Parameter: Decoder $\mathcal{D}$, GPT AR, conditional Model C
Hyper-parameter: multi-scale Resolution $(t_k)_{k=1}^{K}$, interpolation method ϕ

$x_0 = C(T)$
$R = [], S = [x_0]$
for k in $(1, 2, ...K)$ **do**
  $x_k = AR(S)$
  $R = R + x_k$
  if $k = K$ **then**
    break
  else
    $x_k = interpolate(x_k, t_K)$
    $x_k = \phi(interpolate(x_k, t_{k+1}))$
  end if
  $S = S + x_k$
end for
$A = D(R)$
return A

A Supplementary material

A.1 Multi-Scale Residual Quantization

Multi-scale residual quantization (MSRQ) in our SAT is designed to efficiently encode audio signals by leveraging multiple quantization stages on different scales. Specifically, the MSRQ process begins with the raw audio signal being encoded into a feature representation. This representation is then passed through a series of quantizers, each corresponding to a different scale. For each scale, the feature map is downsampled to match the target resolution before quantization. After quantization, the feature map is upsampled back to its original resolution. Due to information loss in interpolation, the upsampled feature map is further processed through our upsampling network to recover the information for each scale. The residual error, calculated after each quantization step, is passed to the next quantizer, allowing the model to refine the audio representation iteratively. The pesudo-code implementation can be shown in Algorithm 1.

A.2 Scale-level Acoustic AutoRegressive Generation.

Our AAR method begins by taking the scale-level tokens generated by the MSRQ process. These tokens are organized hierarchically, with each scale capturing different levels of detail in the audio signal—from coarse, low-frequency information to fine, high-frequency details. In generation, the process is structured to predict these scales sequentially, starting from the coarsest scale and progressing to finer scales. As Algorithm 2 illustrated, our AAR first initializes the generation process by producing an initial latent representation from the input text using a conditional model. This initial representation serves as the starting point for the autoregressive prediction. The model then proceeds through each scale, beginning with the coarsest, and generates the corresponding tokens by conditioning on the sequence of tokens generated thus far. After each scale's tokens are predicted, they are interpolated and refined to align with the resolution of the next finer scale. This iterative process continues until all scales are generated, ensuring a smooth and coherent progression from low to high-frequency details. Finally, the aggregated tokens from all scales are decoded into a complete audio signal, resulting in a high-fidelity output that effectively captures the nuances of the original audio.

A.3 Scale Scheduling

In our paper, we explore three types of scale scheduling: Linear, Quadratic, and Logarithmic.

Scheduler	FAD	STOI	MEL	STFT
Increasing	1.38	0.77	1.39	2.02
Decreasing	1.33	0.74	1.35	1.82

Table 8: Comparison of SAT performance in enlarging and decreasing scales across different metrics.

Specifically, Linear scheduling ensures that the difference between each scale is consistent and linear. For example, in our linear scheduling approach, we start from a scale of 1 and increase to 75 using 16 scales, resulting in a difference of approximately 5 between each consecutive scale. The detail visualization of our scale setting can be shown in Fig. 5.

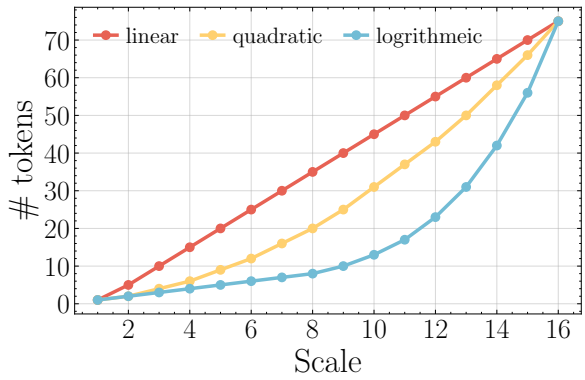


Figure 5: Visualization of Linear, Quadratic, and Logarithmic scale scheduling across the range from 1 to 75.

A.4 Codebook Utilization

We analyze the codebook utilization across different models. In particular, we observe that the codebook utilization for Encodec (Défossez et al., 2022) consistently reaches 99%, indicating that the entire codebook is actively used during the encoding process. In contrast, our SAT model exhibits a lower utilization rate. Specifically, we find that the codebook utilization in every scale of the SAT model remains at approximately 60%. We hypothesize that this discrepancy is caused by the inherent structure of the SAT model, where each time we downsample the input and apply quantization, the model becomes increasingly selective in its use of the codebook entries.

A.5 Token Scale Analysis

To gain more insight into the relationship between the scale scheduler and the SAT performance, we reversed our linear scheduler scales, as illustrated in Tab 8, and observed that the reversed linear scheduler also showed notable results and a higher codebook utilization. We consider that high-frequency information in audio can be sparse, which is the key to such phenomena. However, when we analyze its performance in each scale, we find that because of the decreasing scheduling of the scale, the performance of SAT only dependent on the first scale is still promising, may hinder overall generation performance, thereby contradicting our initial expectations.

A.6 Training Cost Analysis

Our AAR exhibits strong performance in terms of latency and quality while also efficiently reducing the training cost of the model. To be more specific, the vanilla AR achieved an FAD of 8.07 with a classifier-free guidance scale of 4 after training for 100 epochs, while our AAR achieved an improved FAD of 5.70 under the same settings in just 45 epochs. As shown in Tab 9, to achieve the same capacity of vanilla AR, our AAR only needs to train 20 epochs and efficiently save approximately 80% training cost.

Epoch	Method	FAD↓	IS↑	KL↓
100	AR	7.19	2.78	2.73
10	AAR	$7.57_{+0.38}$	$2.97_{+0.19}$	$2.74_{+0.01}$
20	AAR	$6.83_{-0.36}$	$3.24_{+0.46}$	$2.53_{-0.20}$
30	AAR	$6.36_{-0.83}$	$3.49_{+0.71}$	$2.40_{-0.33}$
40	AAR	$6.32_{-0.87}$	$3.55_{+0.77}$	$2.32_{-0.41}$
45	AAR	$6.13_{-1.06}$	$3.63_{+0.85}$	$2.28_{-0.45}$

Table 9: Comparison of training cost between vanilla AR and our AAR. All results are generated within classifier-free guidance scale of 4.

A.7 Upsampling Function

In our work, to efficiently recover information loss from downsampling, we use a 1D convolutional layer after vanilla upsampling to ensure unique information on each scale is preserved and accurately represented. We evaluated its effectiveness through three configurations: unshared (each quantizer has its own convolutional layer); partially shared (approximately three quantizers share one layer); and fully shared (all quantizers use the same layer) to validate the effectiveness of this approach in distinguishing and splitting information across different scales for multi-scale reconstruction, and so on, generation. Our experiments (see Fig. 6) show that the performance of unshared, partially shared, and fully shared networks in reconstruction is similar,

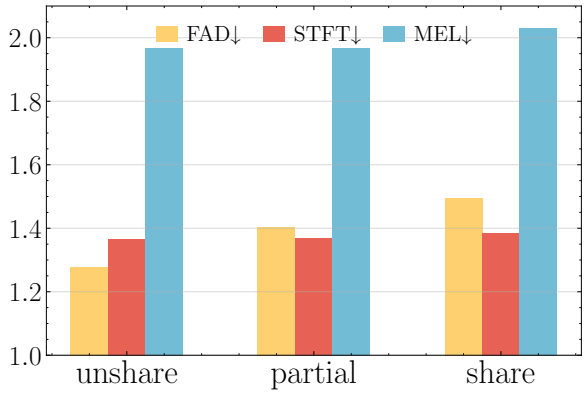


Figure 6: Ablation study of upsampling functions on SAT.

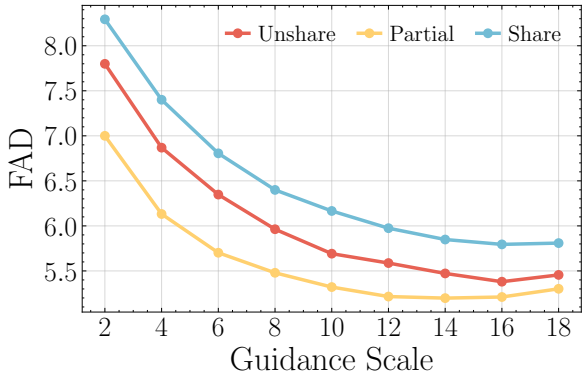


Figure 7: Ablation study of upsampling functions on AAR.

indicating that all configurations effectively maintain audio quality during reconstruction. However, their impact on generation can be seen in Fig. 7, where the partially shared architecture significantly improves generation quality.

B Checklist

This paper:

- Did you describe the limitations of your work? YES
- Did you discuss any potential risks of your work? NA
- Did you use or create scientific artifacts? NO
- Did you run computational experiments? YES
- Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used? YES
- Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values? YES
- Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run? YES
- If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation, such as NLTK, Spacy, ROUGE, etc.), did you report the implementation, model, and parameter settings used? YES
- Did you use human annotators (e.g., crowdworkers) or research with human participants? No
- Did you use AI assistants (e.g., ChatGPT, Copilot) in your research, coding, or writing? YES
- Did you include information about your use of AI assistants? No
- Elaboration For Yes Or No. For yes, provide a section number. For no, justify why not. NA