PyramidCodec: Hierarchical Codec for Long-form Music Generation in Audio Domain

Jianyi Chen[1], Zheqi Dai[2], Zhen Ye[1], Xu Tan[3], Qifeng Liu[1], Yike Guo[1], Wei Xue[1] *

[1]The Hong Kong University of Science and Technology,
[2]The Chinese University of Hong Kong,
[3]Microsoft Research Asia
weixue@ust.hk

Abstract

Generating well-structured long music compositions, spanning several minutes, remains a challenge due to inefficient representation and the lack of structured representation. In this paper, we propose PyramidCodec, a hierarchical discrete representation of audio, for long audio-domain music generation. Specifically, we employ residual vector quantization on different levels of features to obtain the hierarchical discrete representation. The highest level of features has the largest hop size, resulting in the most compact token sequence. The quantized higher-level representation is upsampled and combined with lower-level features to apply residual vector quantization and obtain lower-level discrete representations. Furthermore, we design a hierarchical training strategy to ensure that the details are gradually added with more levels of tokens. By performing hierarchical tokenization, the overall token sequence represents information at various scales, facilitating long-context modeling in music and enabling the generation of well-structured compositions. The experimental results demonstrate that our proposed PyramidCodec achieves competitive performance in terms of reconstruction quality and token per second (TPS). By enabling ultra-long music modeling at the lowest level, the proposed approach facilitates training a language model that can generate well-structured long-form music for up to 3 minutes, whose quality is further demonstrated by subjective and objective evaluations. The samples can be found at `https://pyramidcodec.github.io/`.

1 Introduction

With the advancement of language models, audio generation has achieved significant success, encompassing various domains such as common sound (Borsos et al., 2023; Kreuk et al., 2023; Huang et al., 2023b), speech (Wang et al., 2023; Copet et al., 2024; Zhang et al., 2024; Yang et al., 2023b), and music (Agostinelli et al., 2023; Donahue et al., 2023; Lam et al., 2024; Dhariwal et al., 2020). In these frameworks, audio tokenizers (Zeghidour et al., 2021; Défossez et al., 2022; Yang et al., 2023a; Kumar et al., 2024; Zhang et al., 2024) are employed to obtain discrete representations of audio, which are then used by transformer-based language modeling. In this paper, we focus on music generation in the audio domain, which is challenging due to the complexity and long-form nature of music.

The most dominant idea for audio tokenization is applying vector quantization (VQ). By perfoming VQ in a residual manner, residual vector quantization (RVQ) (Zeghidour et al., 2021) based high-fidelity universal neural audio compression methods, such as Encodec (Défossez et al., 2022), DAC (Kumar et al., 2024), are proposed, which exploit multiple codebooks to gradually capture the acoustic-level audio information from the most important to the least important. In (Yang et al., 2023a), the Hifi-codec is further proposed to balance the information of different levels of codec by using a group-residual vector quantization (GRVQ) method, which divides features into multiple groups and applies RVQ to each group of features. However, to ensure high fidelity, these methods require a large number of codebooks, resulting in excessively long sequences for long-form music. In addition to applying RVQ compression to the audio waveform, CLAM-TTS (Kim et al., 2024) obtains the discrete representation of Mel spectrogram, achieving a high compression rate at approximately 10 Hz. However, to ensure audio fidelity, they still employ 32 codebooks, which leads to a high token per second (TPS) rate.

Although the RVQ enables audio modeling using codebooks at different levels, in fact, they are only focusing on the "acoustic-level" modeling within a fixed time span, rather than "temporal-level" mod-

*Corresponding authors: weixue@ust.hk

eling at different spans. Therefore, regardless of the strategy to improve RVQ, the long-term modeling capacity of conventional methods is limited by the time span for modeling. For instance, some methods (Agostinelli et al., 2023; Borsos et al., 2023; Donahue et al., 2023) generate coarse acoustic tokens (first several RVQ tokens), and the fine acoustic tokens are generated conditioning on coarse acoustic tokens in an auto-regressive manner. We also notice that (Zhang et al., 2024) seeks to integrate semantic information into the first codebook, hoping to generate conceptual outputs. However, for all these methods, tokenizers at all levels are still generated with the same compression rate and token length, hindering the long-term structural modeling of music. It is well-known that music has a hierarchical structure, in which high-level abstractions are the skeleton of music, containing the structure, theme, and rhythm of music, and the low-level information is the detailed high-frequency fluctuations of music. Therefore, to generate structural long music, a hierarchical framework is needed to capture both high-level abstractions and low-level details. Such hierarchical framework has also been applied to image generation such as VQ-VAE-2 (Razavi et al., 2019) and VAR (Tian et al., 2024).

In this paper, we propose PyramidCodec, a hierarchical codec specifically designed generating of well-structured long-form music in the audio domain. PyramidCodec introduces an encoding scheme that operates across multiple scales. By incorporating a hierarchical framework, our codec effectively captures the essence of music at different levels of complexity. At the highest level, PyramidCodec captures the overall structure and essence of the music, producing a temporally compact representation enriched with abstract or skeletal information. As the scale level decreases within PyramidCodec, the temporal resolution of the features increases, enabling the encoding of more intricate and nuanced aspects of the music. RVQ is adopted to quantize the features at all levels, resulting in a multi-scale hierarchical codec.

Following the hierarchical codec design, we further develop a hierarchical training strategy to progressively enhance the codec's ability to reconstruct music at varying scales. Given the ground-truth audio, multi-scale ground-truth targets and loss functions are proposed, following the same idea of hierarchical modeling. The model is trained to perform audio-form reconstruction at different scales. By employing the hierarchical training strategy, we can achieve comparable reconstruction quality at similar compression rates.

Based on the resulting codec, a language model-based abstract-to-detail framework is finally proposed for long-form music generation. The main contributions are summarized as:

- We propose PyramidCodec, a hierarchical discrete representation of audio, specifically designed for long-form music generation. The hierarchical structure allows for efficient representation of both abstractions and details, enabling the generation of well-structured compositions spanning several minutes.
- We introduce a hierarchical training strategy for PyramidCodec, which ensures that the details are gradually added with more levels of tokens. This strategy enables fine reconstruction using full-level codecs and coarse reconstruction by removing some low-level codecs, achieving comparable reconstruction quality at similar compression rates.
- We explore long-form music generation using an abstract-to-detail framework, leveraging PyramidCodec to train an audio language model. This language model is capable of generating well-structured compositions up to 3 minutes in length, whose effectiveness is demonstrated by both objective and subjective evaluations.

2 Related Works

2.1 Audio Tokenizer

Audio tokenizer discretizes audio waveform into tokens, which can be generally divided into two categories: acoustic tokenizer and semantic tokenizer.

Acoustic Tokenizer Acoustic tokenizer aims to achieve a high compression rate while maintaining high reconstruction fidelity. The dominant paradigm for tokenization of image and audio is VQ-VAE (Van Den Oord et al., 2017). This approach has been successfully applied to speech codecs, such as the VQ-VAE based codec proposed in (Gârbacea et al., 2019) operating at 1.6kbps. Other codecs, including Soundstream (Zeghidour et al., 2021), Encodec (Défossez et al., 2022), and DAC (Kumar et al., 2024), also employ residual vector quantization (RVQ) to achieve high-fidelity reconstruction at high TPS. Hifi-codec (Yang et al., 2023a) introduces a group-RVQ technique to balance the information of different levels of the

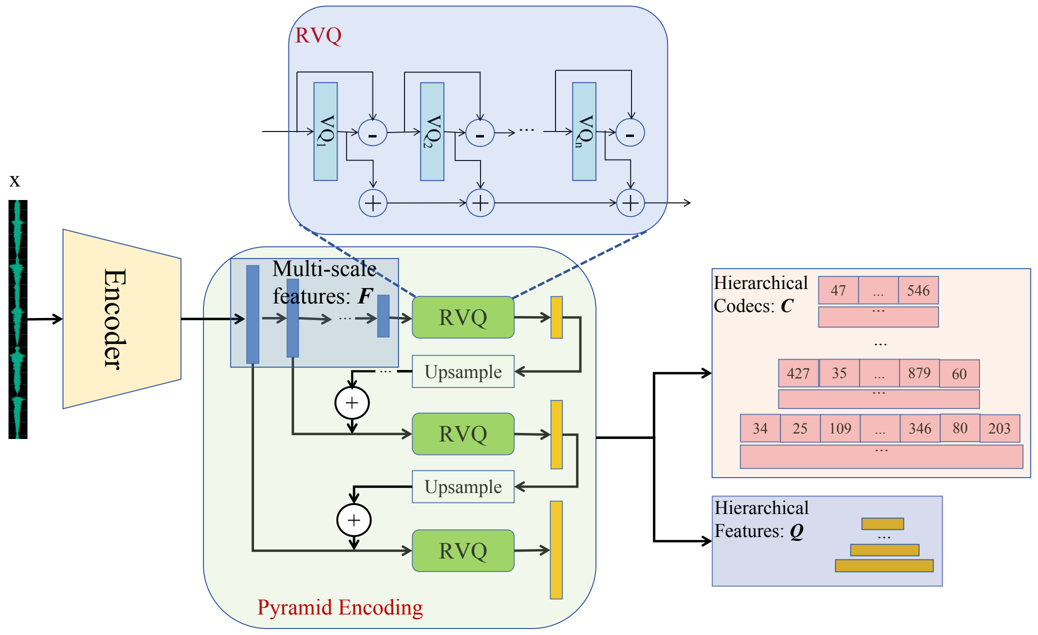

Figure 1: The illustration of pyramid encoding, which takes multi-scale features as input and outputs discrete hierarchical codecs and continuous hierarchical features after quantization. The base feature is extracted through an encoder taking audio waveform as input. Hierarchical codecs is acquired by employing RVQ on multi-scale features. The outputs are discrete hierarchical codecs and continuous hierarchical features after quantization.

codec, operating at considerable TPS. Additionally, scalar quantization technology has been explored in works such as FAQ (Mentzer et al., 2023), (Brendel et al., 2024), and (Ballé et al., 2020).

Semantic Tokenizer: The goal of a semantic tokenizer is to provide discrete tokens with semantic information. These tokens are often used as conditions for acoustic tokens in a two-stage generation framework (Borsos et al., 2023; Agostinelli et al., 2023; Donahue et al., 2023). One approach to acquiring semantic tokens is to use k-means clustering on semantic features extracted from large-pretrained models, such as Hubert (Hsu et al., 2021), w2v-BERT (Chung et al., 2021), and MERT (Li et al., 2023). Another method, proposed by (Zhang et al., 2024), involves obtaining semantic tokens through semantic distillation, which introduces a distillation loss and allows for end-to-end training. (Ye et al., 2024) proposed X-codec which incorporates semantic features from a pre-trained semantic encoder before the Residual Vector Quantization (RVQ) stage and introduces a semantic reconstruction loss after RVQ.

2.2 Music Generation

Music can be represented in two formats: symbolic music and audio music. Symbolic music generation has been extensively studied in works such as (Huang et al., 2018), (Yu et al., 2022), (Hsiao et al., 2021), (Muhamed et al., 2021), and (Mittal et al., 2021). Based on the language model (LM) generation framework, symbolic music is first flattened into tokens, such as MIDI (Huang et al., 2018) or REMI (Huang and Yang, 2020), similar to text. Transformers are then used to predict the next tokens. However, this approach can be highly complex, resulting in extremely long token lengths and potentially broken music structures.

Alternatively, audio-domain music generation, including LM-based methods such as (Agostinelli et al., 2023), (Copet et al., 2024), (Lam et al., 2024), and (Donahue et al., 2023), as well as diffusion-based methods like (Schneider et al., 2023), (Huang et al., 2023a), and (Chen et al., 2024), has achieved great success in recent years. These methods heavily rely on pre-trained high-quality audio codecs in the LM-based approach. Our work provides a kind of hierarchical codec which enables efficient hierarchical music generation.

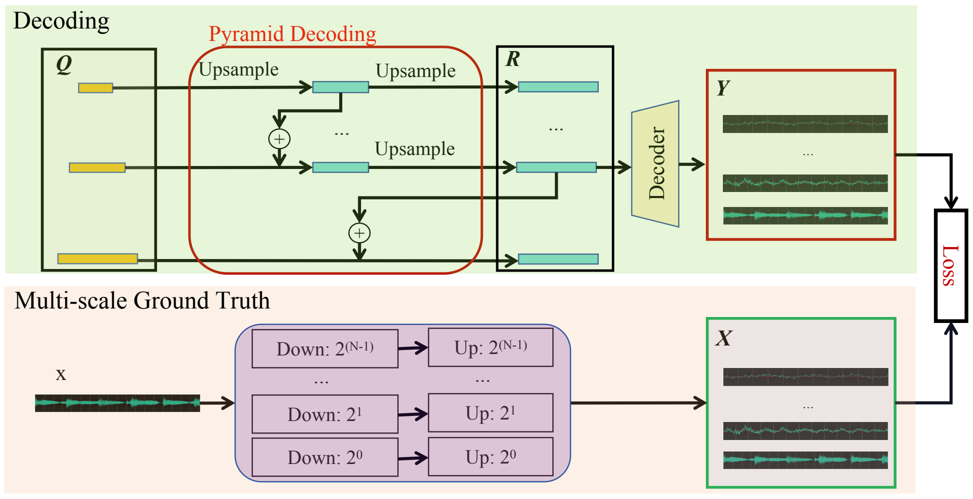

Figure 2: The illustration of pyramid decoding and multi-scale ground truth construction for hierarchical training.

3 Proposed Method

In this section, we propose PyramidCodec, a hierarchical codec designed for efficient long-form music generation. The key problem is how to develop a framework that can jointly learn both high-level abstractions and low-level details, which differs from conventional audio codecs that generally adopt a fixed time span for tokenization at all levels.

The proposed PyramidCodec involves pyramid encoding which gradually constructs the tokens at different time spans along with a series of upsampling, pyramid decoding which reconstructs the audio waveform from the tokens, and a hierarchical training strategy that enforces the details to be added gradually with more levels of tokens. The hierarchical codecs can be used to train an abstract-to-detail language model for music generation. The proposed method is illustrated in Figure2. It is named PyramidCodec since the stack of tokenizers at different levels forms a pyramid structure. Details of different parts are explained as follows.

3.1 Pyramid Encoding

Given input audio, we select a sufficiently long time span and use an encoder, similar to that in DAC, to extract the primary acoustic embedding, which captures the overall information. We then apply a series of down-sampling operations to obtain a set of multi-scale embeddings with different time resolutions. The highest (last) level of embedding represents the most compact information.

To tokenize the embeddings, we start by performing RVQ on the highest level of embedding, resulting in a pair of codec and quantized embedding. We then upsample the quantized embedding and combine it with the next level of embedding. RVQ is performed on the combined embedding, and this process is repeated until the lowest level of embedding is reached. The outputs of the pyramid encoding are hierarchical codecs $\mathbf{C}$ and corresponding quantized hierarchical embeddings $\mathbf{Q}$. The complete process is outlined in Algorithm 1.

Algorithm 1: Pyramid Encoding.

```
1 Pyramid Encoding ($\mathbf{F}$);
  Input  : Multi-scale embedding set $\mathbf{F}$, from
           high level to low level
  Output : Hierarchical codec set $\mathbf{C}$, and
           quantized embedding set $\mathbf{Q}$
  Initial : buffer $B \leftarrow$ None,
           $\mathbf{Q} \leftarrow$ EmptyList(),
           $\mathbf{C} \leftarrow$ EmptyList()
2 for $F$ in $\mathbf{F}$ do
3     $Q, C \leftarrow \mathrm{RVQ}(F + \mathcal{U}(B))$
4     $B \leftarrow Q$
5     $\mathbf{Q}$.append($Q$), $\mathbf{C}$.append($C$)
6 end
  Return : $\mathbf{C}$, $\mathbf{Q}$
```

It is important to note that at each level, the embeddings are upsampled and added back to the next higher-resolution embedding successively. This step is crucial as it reintegrates the quantized residual details back into the primary signal pathway, ensuring that each successive embedding level refines its predecessor. This reintegration helps in recovering the finer temporal details that may have been lost during the initial down-sampling phases.

3.2 Pyramid Decoding

The quantized hierarchical embeddings $\mathbf{Q}$ obtained from pyramid encoding differ in time resolution across different levels. The pyramid decoding aims to transform the embeddings $\mathbf{Q}$ back into a set of new embeddings $\mathbf{R}$, such that at all levels they have the same time resolution as the lowest (first) level of embedding. These resulting embeddings can then be fed into a decoder, similar to the one used in DAC, to reconstruct the audio waveform.

To achieve this, an incremental decoding strategy is proposed, which is expressed by:

$$\mathbf{R}(i) = \mathcal{U}^{N-i-1}[\sum_{k=0}^{i} \mathcal{U}^{i-k}(\mathbf{Q}(i))], \qquad (1)$$

where N is the total number of levels, and $\mathcal{U}^j(x)$ denotes the upsampling of x by j times. This strategy ensures that the embeddings at different levels are mixed in an incremental manner, allowing the coarse audio waveform to be reconstructed by removing some low-level features. The complete process of the pyramid decoding module is outlined in Algorithm 2.

Algorithm 2: Pyramid Decoding.

```
1  Pyramid Decoding ($\mathbf{Q}$);
   Input   : Quantized embedding set $\mathbf{Q}$, from high level to low level
   Output  : Embedding set $\mathbf{R}$ for waveform reconstruction.
   Initial : $\mathbf{R} \leftarrow$ List($\mathbf{Q}[0]$)
2  for $Q$ in $\mathbf{Q}[1:]$ do
3      for $i = 0$ to $len(\mathbf{R})$ do
4          $\mathbf{R} \leftarrow \mathcal{U}(\mathbf{R}[i])$
5      end
6      $F \leftarrow Q + Rs[-1]$
7      $\mathbf{R}$.append($F$)
8  end
   Return : $\mathbf{R}$
```

An illustration of the pyramid decoding is shown in the upper part of Figure 2. It is expected that the resulting embedding set $\mathbf{R}$ will correspond to audio waveforms with varying frequency components.

3.3 Hierarchical Training Strategy

Given the reconstructed audio components at different frequencies, the overall audio waveform is finally generated by gradually adding the signal components from the most compact level to the most detailed level. Specifically, denoting the target audio as X, a set of target signal components $\mathbf{X}$ can be defined, and the i-th level target signal component $\mathbf{X}[i]$ is approximated by $\mathbf{Y}[i]$, based on from $\mathbf{Q}[0]$ to $\mathbf{Q}[i]$. To ensure that all levels of PyramidCodec effectively learn the hierarchical structure of the audio, a hierarchical training strategy is proposed.

Based on the input audio X, the i-th level target signal component $\mathbf{X}[i]$ is defined as

$$\mathbf{X}[i] = \mathcal{U}(\mathcal{D}(X, 2^{N-i}), 2^{N-i}) \qquad (2)$$

where N is the total number of scales, $\mathcal{D}(x, r)$ denotes the down-sampling of x by a factor of r, and $\mathcal{U}(x, r)$ denotes the up-sampling of x by a factor of r. We note that by performing downsampling and upsampling, the target signal components $\mathbf{X}$ are obtained at different frequencies.

Finally, the hierarchical training losses L_s are defined as follows:

$$L_s[i] = \text{loss}(Y_s[i], \mathbf{X}[i]), \forall i = 0, 1, ..., N-1 \qquad (3)$$

In our approach, we utilize the Mel spectrogram loss and discriminator loss, similar to the DAC framework (Kumar et al., 2024).

3.4 Abstract-to-Detail Music Generation

In this section, we propose an abstract-to-detail framework to generate long-term audio music.

In the proposed framework, the PyramidCodec is used to encode abstraction information into a compact token sequence, which represents the structure, theme, and rhythm of the music. The generation starts by producing the abstraction tokens, which define the basic structure of the music. Then, low-level tokens, which include more intricate and nuanced details such as chords and high-frequency components, are generated conditioned on the abstraction tokens. This incremental generation process continues by generating lower-level tokens conditioned on all higher-level tokens. We flatten all the tokens and finally train a transformer-based language model for music generation.

With hierarchical modeling, we can control the level of detail in the generated music by using different numbers of high-level tokens, providing a trade-off between computational complexity and quality. Even with the highest-level tokens alone, at a token per second (TPS) rate of 10, the generated music still exhibits a coherent structure and harmonious rhythms.

Model	Mel$_{dis}$(↓)	STFT$_{dis}$(↓)	Waveform$_{dis}$(↓)	SI-SDR(↑)	TPS(↓)
DAC-16k-n12-q12	1.096	1.334	0.021	-8.662	600
DAC-16k-n12-q2	2.419	2.009	0.020	-12.944	100
Encodec-24k-n8-q8	1.552	2.288	0.009	4.319	600
Encodec-24k-n8-q2	1.938	2.458	0.015	-1.722	150
DAC-22k-n9-q9 (*)	0.756	1.149	0.004	12.277	388
Pyramid-22k-p555 (*)	0.811	1.235	0.005	10.142	350
DAC-22k-n2-q2 (*)	0.929	1.204	0.007	7.153	86
Pyramid-22k-p111 (*)	0.953	1.247	0.008	6.613	70

Table 1: Results on reconstruction evaluation. It includes randomly selected 10,000 10-second audio samples from our testing dataset. Models trained on our dataset are denoted with an asterisk (*). The notation follows a standard rule, such as DAC-16k-n12-q2 for DAC pretrained models operating at a 16,000 sampling rate with 12 codebooks (n12), using only the first 2 codebooks for reconstruction (q2). Pyramid models are represented by notations like Pyramid-22k-p111, indicating a 22,050 sampling rate and three levels with one codebook per level (p111).

4 Experiments

4.1 Dataset

We use the Lakh MIDI (LMD) dataset (Raffel, 2016) for both the PyramidCodec and music generation experiments. The LMD dataset consists of approximately 160,000 multi-track MIDI music files. To simplify the dataset, we merge similar tracks into a single basic track, resulting in five tracks: square synthesizer, piano, guitar, string, bass, and drum. This preprocessing follows (Yu et al., 2022) and (Ren et al., 2020).

Further, we utilized FluidSynth [1] and the soundFont 2 sound bank file **TimGM6mb.sf2** from the pretty-midi library [2] to synthesize the audio wave files. The resulting dataset consists of approximately 5,000 hours of audio files, all sampled at a rate of 22,050Hz. We randomly split the audio dataset into training, validation, and testing sets with a ratio of 8:1:1, respectively.

4.2 PyramidCodec Evaluation

4.2.1 Implementation and Baselines

The encoder, which is used to obtain the primary acoustic embedding in Section 3.1, and the decoder, which is used to recover the audio waveform, are similar to the DAC implementation [3]. We set the encoder rates and decoder rates to be [4, 4, 4, 4, 2] and [2, 4, 4, 4, 4], respectively, resulting in a stride length of 512. For all experiments, the sampling rate is set to 22,050 Hz. The codebook size is set to 1024, the same as DAC. Our approach utilizes a single decoder for audio waveform decoding, with different scales of tokens being selectively integrated through a lightweight pyramid decoding module. The model size and inference speed are comparable to those of DAC. To prevent the codebook from collapsing, we adjust the weight of the Mel loss to have less or more impact relative to the codebook loss. During the warming-up stage, the weight of the Mel loss is set to 2.0, and later it is increased to 10.0.

In our implementation, the up-sampling operation used in Algorithm 1 and Algorithm 2 is performed using the learnable "ConvTranspose1d" [4] layer. On the other hand, the down-sampling and up-sampling operations described in Section 3.3 for constructing multi-scale ground truth utilize "julius.resample_frac" [5], which is based on Julius O. Smith's resampling algorithm. This algorithm, often referred to as the "polyphase filter bank" approach, is a well-established method in digital signal processing for changing the sample rate of a signal. It is known for its efficiency and high-quality results.

There are high-fidelity pre-trained codecs trained on large datasets, which are used by many generation works. We have designed two types of baselines: weak baselines and strong baselines. For the weak baselines, we use the pre-trained Encodec [6] and DAC [3] as they were not trained on our dataset. For the strong baselines, we trained a revised version of DAC from scratch on our dataset with two settings: 1) high-fidelity setting, where the number of codebooks is 9; 2) efficient setting, where the

[1]https://www.fluidsynth.org/

[2]https://github.com/craffel/pretty-midi/tree/main/pretty-midi

[3]https://github.com/descriptinc/descript-audio-codec

[4]https://pytorch.org/docs/2.2/generated/torch.nn.ConvTranspose1d

[5]https://adefossez.github.io/julius/julius/resample.html

[6]https://github.com/facebookresearch/encodec

number of codebooks is 2.

4.2.2 Evaluation Metrics

We evaluate the reconstruction quality of our models and baseline model using the following objective metrics:

Mel distance: The L1 distance between the Mel spectrograms with window lengths of [32, 64, 128, 256, 512, 1024, 2048], a hop size equal to a quarter of the window length, and the number of Mel frequency banks set to [5, 10, 20, 40, 80, 160, 320].

STFT distance: The L1 distance between the log magnitude spectrograms with window lengths of [2048, 512].

Waveform distance: The L1 distance between the ground truth audio waveforms and the reconstructed ones.

Scale-invariant source-to-distortion ratio (SI-SDR) (Le Roux et al., 2019): This metric provides a signal-to-noise ratio that is invariant to scale differences, indicating the quality of phase reconstruction compared to spectral metrics.

Tokens per second (TPS): We calculate the number of tokens per second, which indicates the compactness of tokens and is essential for the language model.

4.2.3 Results

Reconstruction Quality. Comparison results are shown in Table 1. Existing codecs, such as DAC-16k and Encodec-14k, achieve high TPS values reaching 600. However, for our goal of performing long-sequence music generation, a lower TPS is preferred. When decreasing the TPS, we observe a considerable degradation in reconstruction quality.

To improve the performance of DAC, we train it on the dataset described in Section 4.1. We find that there is a trade-off between audio quality and TPS in DAC, where increasing the TPS leads to better audio quality. We also implement the PyramidCodec to approximate the TPS of DAC by a proper choice of TPS at each level. We observe that with lower TPS, the PyramidCodec achieves slightly worse reconstruction quality compared to DAC, indicating that the reconstruction performance of PyramidCodec is actually comparable to DAC. However, as will be shown later, the proposed PyramidCodc could generate better music pieces than conventional codecs.

We specifically note that in order to perform long-term audio generation, two efficient codecs, `DAC-22k-n2-q2` at 86 TPS and

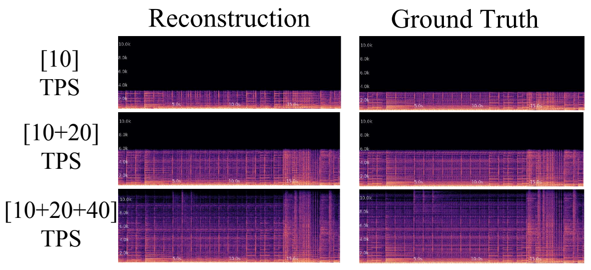

Figure 3: Hierarchical reconstruction results.

`Pyramid-22k-p111` at 70 TPS are trained, and the music generation experiments will be conducted on these two codecs.

Hierarchical Reconstruction. Here we further illustrate the intermediate reconstruction results of PyramidCodec. For `Pyramid-22k-p111`, the TPS can be 70, 30, and 10 when using all levels, the first two levels, and the first level of codes, respectively. The reconstruction Mel spectrograms are shown in Figure 3. The highest level of code is responsible for a quarter of the low-frequency components, while half of the frequency components are reconstructed using the first two levels of codes. The full frequency components are reconstructed using all levels of codes.

4.2.4 Discussion about the Number of Levels

We fixed the number of levels in our model at 3 to balance complexity and quality, as discussed in Table 1, with levels operating at 10 TPS (highest) and 40 TPS (lowest). While we also tested 5 levels ([5, 10, 20, 40, 80] TPS), we found that the highest level (5 TPS) lacked information and the lowest (80 TPS) was inefficient for long-form generation. Thus, three levels were chosen. Reconstruction quality for partial levels showed Mel distances of 0.66, 0.89, and 0.95 for the first level, first two levels, and all levels, respectively, as described in Section 3.3. The ground truth of each level for these measurements is described in Equation 2 in Section 3.3.

4.3 Music Generation Evaluation

We employ GPT-2 (Radford et al., 2019) as the underlying language model for our music generation experiments. These experiments are conducted using two different codecs: the DAC `DAC-22k-n2-q2` with 86 TPS and the Pyramid-Codec `Pyramid-22k-p111` with 70 TPS, which represent the coarse-to-fine and abstract-to-detail generation frameworks, respectively. We will denote the two methods as "DAC-TPS86" and

Model	FAD$_{\text{VGGish}}$(↓)	FAD$_{\text{MERT}}$(↓)	FAD$_{\text{MERT-11}}$(↓)	FAD$_{\text{CLAP-MUSIC}}$(↓)	FAD$_{\text{CLAP-AUDIO}}$(↓)	FAD$_{\text{Encodec}}$(↓)
DAC-TPS86$_{t=1.0}$	0.1814	2.1496	1.2682	0.0271	0.0348	1.4392
Pyramid-TPS70$_{t=1.0}$	0.1678	2.1041	**1.0308**	0.0309	0.0403	1.0914
DAC-TPS86$_{t=1.3}$	0.2609	2.3116	1.2239	0.0306	0.0339	1.0931
Pyramid-TPS70$_{t=1.3}$	0.1576	**2.0936**	1.0856	**0.0257**	**0.0319**	**1.0763**
DAC-TPS86$_{t=0.8}$	0.2811	2.2511	1.2281	0.0546	0.0643	3.5865
Pyramid-TPS70$_{t=0.8}$	**0.1518**	2.4011	1.4553	0.0346	0.0428	1.2259
AudioLM(Encodec)	1.6642	9.5679	5.0767	0.1823	0.2163	7.2886

Table 2: Objective evaluation on music continuation. We randomly pick 500 samples from the testing set. DAC-TPS86 is re-trained on our dataset with two codebooks. t=1.0 means the sampling temperature is 1.0.

"PyramidCodc-TPS70" respectively. Additionally, we have implemented AudioLM (Borsos et al., 2023) as a baseline for music continuation.

4.3.1 Implementation and Baselines

We train our language model using the codebase nanoGPT [7]. Our model consists of 6 layers, each with 8 attention heads, and an embedding dimension of 768. We modify the original vocabulary size to 1028, with 1024 tokens for the codebook and 4 tokens for special purposes. The total number of trainable parameters in our model is approximately 40 million.

The maximum audio duration of training samples is 163 seconds (14,000 tokens) for the DAC-TPS86 and 200 seconds (14,000 tokens) for the PyramidCodc-TPS70 framework. The training is performed on a single 32G V100 GPU with a batch size of 6 for approximately 500,000 steps. All models are trained using bfloat16 precision.

We follow the codebase[8] to train AudioLM. The pre-trained Encodec-16k is used as the tokenizer, and we follow the instructions to train the semantic model, coarse model, and fine model. The length of the audio waveform is set to 15 seconds, and each model is trained on a single 32G V100 GPU.

Two types of tasks are designed: 1) Music continuation: Given the previous 10 seconds as a prompt, the task is to compose the remaining 30 seconds of music. 2) Long-form music generation: The task is to generate 3 minutes of music from scratch.

4.3.2 Evaluation Metrics

Objective Evaluation. We follow (Chen et al., 2024) and (Gui et al., 2024) to use FAD as objective metrics for music evaluation. Here, we incorporate three types of embedding extractors: VGGish (Hershey et al., 2017), MERT (Li et al., 2023), clap-laion (Wu et al., 2023), and Encodec (Défossez et al., 2022). This results in the metrics FAD$_{\text{VGGish}}$, FAD$_{\text{MERT}}$, FAD$_{\text{CLAP-MUSIC}}$, FAD$_{\text{CLAP-AUDIO}}$, and FAD$_{\text{Encodec}}$.

Subjective Evaluation. To assess the quality of the generated music, we employ Mean Opinion Score (MOS) through human evaluation. We use MTurk[9] for subjective evaluation. For music continuation, we ask the testers to evaluate the overall quality of 200 samples for each method. Each sample is evaluated by 2 testers. For long-form music generation, we evaluate the music subjectively on 50 samples for each method in three dimensions: coherence of music structure, rhythms, and music convention. Each sample is evaluated by 3 testers.

[7]https://github.com/karpathy/nanoGPT

[8]https://github.com/lucidrains/audiolm-pytorch

Model	**MOS**(↑)
Ground Truth	4.18
DAC-TPS86	3.35
PyramidCodec-TPS70	**3.47**
AudioLM	2.43

Table 3: Subjective evaluation on music continuation. All samples are generated with temperature 1.0.

Model	**Coherence**(↑)	**Rhythms**(↑)	**Convention**(↑)
Ground Truth	4.11	3.71	4.43
DAC-TPS86	3.25	2.52	3.63
PyramidCodec-TPS70	**3.43**	**2.97**	**3.83**

Table 4: Subjective evaluation of long-form music generation from scratch. All samples are generated with temperature 1.0.

4.3.3 Results

Music Continuation. The results are shown in Table 2. Continuing to compose 30 seconds of music based on a 10-second prompt requires the model to handle at least 40-second audio windows. However, AudioLM (Borsos et al., 2023), which employs an inefficient codec and is trained with a window size of 15 seconds, requires multiple rounds to generate 30 seconds of music, with 15 seconds for each round. As shown in Table 2, AudioLM performs

[9]https://www.mturk.com/

poorly in the 30-second continuation task due to its inefficient coarse-to-fine generation process.

On the other hand, both DAC and PyramidCodec are efficient codecs trained on our dataset. Among them, PyramidCodec generally performs better across different sampling temperature parameters. The objective evaluation results, shown in Table 3, indicate that PyramidCodec outperforms AudioLM, while DAC performs comparably. These results are consistent with the subjective evaluation, where PyramidCodec is rated the highest and AudioLM is rated the lowest.

In summary, both objective and subjective evaluations demonstrate the superiority of the proposed framework for music continuation.

Long-form Music Generation. Our two efficient codecs can handle 3 minutes of music within a single model during both the training and inference stages. We generated 3 minutes of music from scratch using the re-trained DAC-TPS86 and PyramidCodec-TPS70. Table 4 presents the subjective evaluation results, indicating that the music generated with the PyramidCodec outperforms DAC in terms of coherence of music structure, rhythms, and adherence to musical conventions.

5 Conclusion

In this paper, we introduce PyramidCodec, a hierarchical codec designed for long-form music generation in the audio domain. We propose a hierarchical training strategy for PyramidCodec, where details are progressively incorporated using multiple levels of tokens. Through PyramidCodec, we investigate the abstract-to-detail generation framework for music, and our experimental results highlight its effectiveness.

In future work, we plan to explore the performance of PyramidCodec and the abstract-to-detail generation framework on real audio data, including speech, singing, music, and audio effects. Additionally, we aim to explore conditional generation tasks, such as text to music.

6 Limitation

Our study, while providing an insight for long-form music generation, has several limitations. Firstly, our method has been solely validated using synthetic music datasets obtained from MIDI. It is important to recognize that real complex music may require a more extensive representation, which could potentially impede long-form generation. Secondly, we have not validated the effectiveness on other types of audio, such as speech and audio effects. Thirdly, we have not yet explored the domain of multi-modal long music generation, which involves incorporating multiple modalities such as text-to-music.

Acknowledgments

The research was supported by Early Career Scheme (ECS-HKUST22201322), Theme-based Research Scheme (T45-205/21-N) from Hong Kong RGC, NSFC (No. 62206234), and Generative AI Research and Development Centre from InnoHK.

References

Andrea Agostinelli, Timo I Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Caillon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, et al. 2023. Musiclm: Generating music from text. *arXiv preprint arXiv:2301.11325*.

Johannes Ballé, Philip A Chou, David Minnen, Saurabh Singh, Nick Johnston, Eirikur Agustsson, Sung Jin Hwang, and George Toderici. 2020. Nonlinear transform coding. *IEEE Journal of Selected Topics in Signal Processing*.

Zalán Borsos, Raphaël Marinier, Damien Vincent, Eugene Kharitonov, Olivier Pietquin, Matt Sharifi, Dominik Roblek, Olivier Teboul, David Grangier, Marco Tagliasacchi, et al. 2023. Audiolm: a language modeling approach to audio generation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*.

Andreas Brendel, Nicola Pia, Kishan Gupta, Guillaume Fuchs, and Markus Multrus. 2024. Simple and efficient quantization techniques for neural speech coding. *arXiv preprint arXiv:2405.08417*.

Jianyi Chen, Wei Xue, Xu Tan, Zhen Ye, Qifeng Liu, and Yike Guo. 2024. FastSAG: Towards fast non-autoregressive singing accompaniment generation. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Yu-An Chung, Yu Zhang, Wei Han, Chung-Cheng Chiu, James Qin, Ruoming Pang, and Yonghui Wu. 2021. W2v-bert: Combining contrastive learning and masked language modeling for self-supervised speech pre-training. In *Proc. IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*.

Jade Copet, Felix Kreuk, Itai Gat, Tal Remez, David Kant, Gabriel Synnaeve, Yossi Adi, and Alexandre Défossez. 2024. Simple and controllable music generation. In *Proc. Conf. Neural Information Processing Systems (NeurIPS)*.

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

Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. 2020. Jukebox: A generative model for music. *arXiv preprint arXiv:2005.00341*.

Chris Donahue, Antoine Caillon, Adam Roberts, Ethan Manilow, Philippe Esling, Andrea Agostinelli, Mauro Verzetti, Ian Simon, Olivier Pietquin, Neil Zeghidour, et al. 2023. Singsong: Generating musical accompaniments from singing. *arXiv preprint arXiv:2301.12662*.

Cristina Gârbacea, Aäron van den Oord, Yazhe Li, Felicia SC Lim, Alejandro Luebs, Oriol Vinyals, and Thomas C Walters. 2019. Low bit-rate speech coding with vq-vae and a wavenet decoder. In *Proc. IEEE Intl. Conf. Acoustics, Speech, Signal Process. (ICASSP)*.

Azalea Gui, Hannes Gamper, Sebastian Braun, and Dimitra Emmanouilidou. 2024. Adapting frechet audio distance for generative music evaluation. In *Proc. IEEE Intl. Conf. Acoustics, Speech, Signal Process. (ICASSP)*.

Shawn Hershey, Sourish Chaudhuri, Daniel PW Ellis, Jort F Gemmeke, Aren Jansen, R Channing Moore, Manoj Plakal, Devin Platt, Rif A Saurous, Bryan Seybold, et al. 2017. Cnn architectures for large-scale audio classification. In *Proc. IEEE Intl. Conf. Acoustics, Speech, Signal Process. (ICASSP)*.

Wen-Yi Hsiao, Jen-Yu Liu, Yin-Cheng Yeh, and Yi-Hsuan Yang. 2021. Compound word transformer: Learning to compose full-song music over dynamic directed hypergraphs. In *Proc. AAAI Conf. Artif. Intell. (AAAI)*.

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 Trans. Audio, Speech, Lang. Process.*

Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Ian Simon, Curtis Hawthorne, Andrew M Dai, Matthew D Hoffman, Monica Dinculescu, and Douglas Eck. 2018. Music transformer. *arXiv preprint arXiv:1809.04281*.

Qingqing Huang, Daniel S Park, Tao Wang, Timo I Denk, Andy Ly, Nanxin Chen, Zhengdong Zhang, Zhishuai Zhang, Jiahui Yu, Christian Frank, et al. 2023a. Noise2music: Text-conditioned music generation with diffusion models. *arXiv preprint arXiv:2302.03917*.

Rongjie Huang, Jiawei Huang, Dongchao Yang, Yi Ren, Luping Liu, Mingze Li, Zhenhui Ye, Jinglin Liu, Xiang Yin, and Zhou Zhao. 2023b. Make-an-audio: Text-to-audio generation with prompt-enhanced diffusion models. In *Proc. Intl. Conf. Machine Learning (ICML)*.

Yu-Siang Huang and Yi-Hsuan Yang. 2020. Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions. In *Proc. ACM Multimedia (ACM MM)*.

Jaehyeon Kim, Keon Lee, Seungjun Chung, and Jaewoong Cho. 2024. Clam-tts: Improving neural codec language model for zero-shot text-to-speech. *arXiv preprint arXiv:2404.02781*.

Felix Kreuk, Gabriel Synnaeve, Adam Polyak, Uriel Singer, Alexandre Défossez, Jade Copet, Devi Parikh, Yaniv Taigman, and Yossi Adi. 2023. Audiogen: Textually guided audio generation. In *Proc. Intl. Conf. Learning Representations (ICLR)*.

Rithesh Kumar, Prem Seetharaman, Alejandro Luebs, Ishaan Kumar, and Kundan Kumar. 2024. High-fidelity audio compression with improved rvqgan. In *Proc. Conf. Neural Information Processing Systems (NeurIPS)*.

Max WY Lam, Qiao Tian, Tang Li, Zongyu Yin, Siyuan Feng, Ming Tu, Yuliang Ji, Rui Xia, Mingbo Ma, Xuchen Song, et al. 2024. Efficient neural music generation. In *Proc. Conf. Neural Information Processing Systems (NeurIPS)*.

Jonathan Le Roux, Scott Wisdom, Hakan Erdogan, and John R Hershey. 2019. Sdr–half-baked or well done? In *Proc. IEEE Intl. Conf. Acoustics, Speech, Signal Process. (ICASSP)*.

Yizhi Li, Ruibin Yuan, Ge Zhang, Yinghao Ma, Xingran Chen, Hanzhi Yin, Chenghua Lin, Anton Ragni, Emmanouil Benetos, Norbert Gyenge, et al. 2023. Mert: Acoustic music understanding model with large-scale self-supervised training. *arXiv preprint arXiv:2306.00107*.

Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. 2023. Finite scalar quantization: Vq-vae made simple. *arXiv preprint arXiv:2309.15505*.

Gautam Mittal, Jesse Engel, Curtis Hawthorne, and Ian Simon. 2021. Symbolic music generation with diffusion models. In *Proc. Intl. Soc. for Music Information Retrieval Conf. (ISMIR)*.

Aashiq Muhamed, Liang Li, Xingjian Shi, Suri Yaddanapudi, Wayne Chi, Dylan Jackson, Rahul Suresh, Zachary C Lipton, and Alex J Smola. 2021. Symbolic music generation with transformer-gans. In *Proc. AAAI Conf. Artif. Intell. (AAAI)*.

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

Colin Raffel. 2016. *Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching*. Columbia University.

Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. 2019. Generating diverse high-fidelity images with vq-vae-2. In *Proc. Conf. Neural Information Processing Systems (NeurIPS)*.

Yi Ren, Jinzheng He, Xu Tan, Tao Qin, Zhou Zhao, and Tie-Yan Liu. 2020. Popmag: Pop music accompaniment generation. In *Proc. ACM Multimedia (ACM MM)*.

Flavio Schneider, Ojasv Kamal, Zhijing Jin, and Bernhard Schölkopf. 2023. Mo\^ usai: Text-to-music generation with long-context latent diffusion. *arXiv preprint arXiv:2301.11757*.

Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. 2024. Visual autoregressive modeling: Scalable image generation via next-scale prediction. *arXiv preprint arXiv:2404.02905*.

Aaron Van Den Oord, Oriol Vinyals, et al. 2017. Neural discrete representation learning.

Chengyi Wang, Sanyuan Chen, Yu Wu, Ziqiang Zhang, Long Zhou, Shujie Liu, Zhuo Chen, Yanqing Liu, Huaming Wang, Jinyu Li, et al. 2023. Neural codec language models are zero-shot text to speech synthesizers. *arXiv preprint arXiv:2301.02111*.

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 *Proc. IEEE Intl. Conf. Acoustics, Speech, Signal Process. (ICASSP)*.

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

Dongchao Yang, Songxiang Liu, Rongjie Huang, Chao Weng, and Helen Meng. 2023b. Instructtts: Modelling expressive tts in discrete latent space with natural language style prompt. *arXiv preprint arXiv:2301.13662*.

Zhen Ye, Peiwen Sun, Jiahe Lei, Hongzhan Lin, Xu Tan, Zheqi Dai, Qiuqiang Kong, Jianyi Chen, Jiahao Pan, Qifeng Liu, et al. 2024. Codec does matter: Exploring the semantic shortcoming of codec for audio language model. *arXiv preprint arXiv:2408.17175*.

Botao Yu, Peiling Lu, Rui Wang, Wei Hu, Xu Tan, Wei Ye, Shikun Zhang, Tao Qin, and Tie-Yan Liu. 2022. Museformer: Transformer with fine-and coarse-grained attention for music generation. In *Proc. Conf. Neural Information Processing Systems (NeurIPS)*.

Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. 2021. Soundstream: An end-to-end neural audio codec. *IEEE/ACM Trans. Audio, Speech, Lang. Process.*

Xin Zhang, Dong Zhang, Shimin Li, Yaqian Zhou, and Xipeng Qiu. 2024. Speechtokenizer: Unified speech tokenizer for speech language models. In *Proc. Intl. Conf. Learning Representations (ICLR)*.