SUNAC: Source-aware Unified Neural Audio Codec

Aihara, Ryo; Masuyama, Yoshiki; Paissan, Francesco; Germain, François G; Wichern, Gordon; Le Roux, Jonathan

TR2026-032 March 19, 2026

Abstract

Neural audio codecs (NACs) provide compact representations that can be leveraged in many downstream applications, in particular large language models. Yet most NACs encode mixtures of multiple sources in an entangled manner, which may impede efficient down- stream processing in applications that need access to only a subset of the sources (e.g., analysis of a particular type of sound, transcription of a given speaker, etc). To address this, we propose a source-aware codec that encodes individual sources directly from mixtures, conditioned on source type prompts. This enables user-driven selection of which source(s) to encode, including separately encoding multiple sources of the same type (e.g., multiple speech signals). Ex- periments show that our model achieves competitive resynthesis and separation quality relative to a cascade of source separation followed by a conventional NAC, with lower computational cost.

IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP) 2026

SUNAC: SOURCE-AWARE UNIFIED NEURAL AUDIO CODEC

Ryo Aihara[1,2], *Yoshiki Masuyama*[1], *Francesco Paissan*[1,3],
François G. Germain[1], *Gordon Wichern*[1], *Jonathan Le Roux*[1]

[1]Mitsubishi Electric Research Laboratories (MERL), Cambridge, USA
[2]Mitsubishi Electric Corporation, Japan
[3]University of Trento, Trento, Italy

ABSTRACT

Neural audio codecs (NACs) provide compact representations that can be leveraged in many downstream applications, in particular large language models. Yet most NACs encode mixtures of multiple sources in an entangled manner, which may impede efficient downstream processing in applications that need access to only a subset of the sources (e.g., analysis of a particular type of sound, transcription of a given speaker, etc). To address this, we propose a source-aware codec that encodes individual sources directly from mixtures, conditioned on source type prompts. This enables user-driven selection of which source(s) to encode, including separately encoding multiple sources of the same type (e.g., multiple speech signals). Experiments show that our model achieves competitive resynthesis and separation quality relative to a cascade of source separation followed by a conventional NAC, with lower computational cost.

Index Terms— neural audio codecs, source separation, speech enhancement, prompting, source-aware

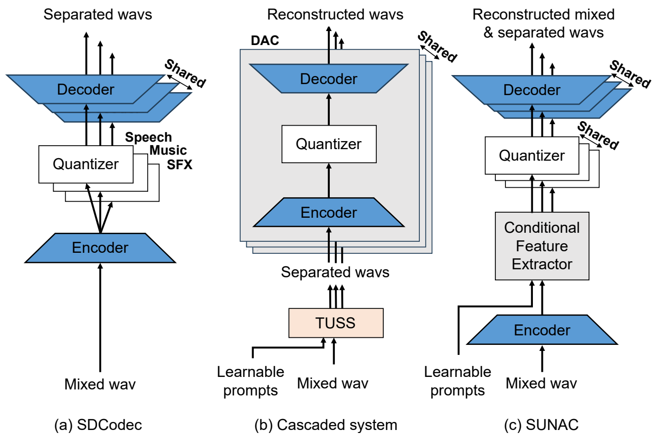

Fig. 1. Overview of (a) SDCodec [20], (b) the proposed cascaded system, and (c) the proposed SUNAC model.

1. INTRODUCTION

With the advent of large language models, end-to-end discrete neural audio codecs (NACs) have been widely investigated as a front end for converting audio signals into discrete text-like tokens [1, 2]. A typical setup employs generative adversarial network (GAN)-based training, a convolutional encoder for waveform analysis, a residual vector quantization (RVQ) module for discretization, and a (transposed-)convolutional decoder for waveform synthesis [3–6].

Because speech communication over a channel is a primary application of NACs, their noise robustness has received considerable attention [7–10]. In real-world scenarios, however, received audio often contains multiple concurrent sources, such as speech, music, and environmental sounds. Most conventional NACs are trained without source awareness, and thus encode mixtures without disentangling the constituent sources. We posit that encoding and transmitting mixtures in this manner is suboptimal for downstream tasks that predominantly target a single source (e.g., meeting summarization [11, 12], full-duplex voice assistants [13–15], acoustic event detection [16, 17], music-language models [18, 19], etc).

SDCodec [20] addressed this challenge by augmenting conventional NACs with parallel, source-aware RVQ modules (Fig. 1(a)). This straightforward design can separately encode and reconstruct sources from mixtures of speech, music, and sound effects (SFX). However, because the separation capacity is tied to distinct source-aware RVQs, the method cannot separate mixtures of sources from the same category (e.g., two concurrent speakers).

This work was done while F. Paissan was an intern at MERL.

To separately encode sources (including those from the same category), we first propose a cascaded architecture that couples a unified source separator [21–23] with a conventional NAC, i.e., first separate, then encode (Fig. 1(b)). For the separator, we use the recently introduced task-aware unified source separator (TUSS) [24], which consolidates multiple separation tasks within a single model and selects the desired operation via lightweight prompting. However, this cascaded design is not computationally efficient, since both the separator and the NAC independently derive compact representations from the same input audio, leading to redundant computation.

We thus also propose a source-aware unified neural audio codec (SUNAC) (Fig. 1(c)), where all components are trained end-to-end with joint optimization, in contrast to the cascaded system. SUNAC performs prompt-based source feature extraction directly in the latent space, after which a quantizer estimates codes on the separated features. To address permutation ambiguity when processing multiple sources of the same type, we apply permutation invariant training (PIT) [25, 26] in the feature space. Both the cascaded model and SUNAC can separate multiple sources from the same or different types. Moreover, the prompting mechanism removes any predefined cap on the number of sources, allowing the model to scale to an arbitrary number in principle. Experimental results show that SUNAC is comparable to SDCodec in scenarios without multiple sources from the same category, that it can also encode and reconstruct each speech source in multi-speaker mixtures. Furthermore, SUNAC achieves performance on par with the cascaded architecture while offering lower computational cost.

2. SOURCE-AWARE UNIFIED NEURAL AUDIO CODEC

2.1. Problem Setup

We consider an input waveform $\mathbf{x} = \sum_n \mathbf{s}_n^{(p_n)} \in \mathbb{R}^L$ consisting of $N \geq 1$ sources $\mathbf{s}_n^{(p_n)}$, each associated with a source type described by a prompt $p_n \in \mathcal{T}$. In this paper, we consider sources to be either speech, music, SFX, or a mixture of these, such that $\mathcal{T} = \{$`<Speech>`, `<Music>`, `<SFX>`, `<Mix>`$\}$. Multiple sources may be of the same type. Our goal is to extract codes for one or more of these sources, specified by a set of desired prompts.

2.2. SDCodec

SDCodec [20], illustrated in Fig. 1(a), considers a restricted version of our setup where $N = 3$ and there is one source each of speech, music, and SFX. It extends the Descript audio codec (DAC) [5] to handle parallel processing of mixture components via the insertion of source-aware RVQ modules after the convolutional encoder. This design compels all sources to share a common encoder-derived feature space, while enabling per-source quantization pathways; consequently, orthogonality across source features is encouraged, similarly to the approach in [8]. By contrast, our proposed SUNAC estimates a prompt-conditioned feature space, removing the need to enforce such orthogonality. Moreover, SDCodec reconstructs mixtures by decoding the sum of the per-source quantized features, imposing additivity in the quantized space, whereas SUNAC reconstructs mixtures directly by prompting the model with a `<Mix>` prompt. Finally, SDCodec cannot separate multiple sources of the same type.

2.3. Separation and NAC Cascade

As a straightforward approach for our problem setup, we propose to consider a cascaded system which first separates the mixture into the desired source-specific waveform signals, before encoding them. A natural choice is to employ TUSS [24] as a front end to DAC, as illustrated in Fig. 1(b). In TUSS, the mixture waveform is encoded via short-time Fourier transform (STFT) and a band-split encoder. The encoded features and learnable prompts indicating the target source type are transformed by TF-Locoformer layers [27]. The transformed features are conditioned by element-wise multiplication with the transformed prompts, after appropriate broadcast. The conditioned features are then refined by more TF-Locoformer layers and mapped back to the time domain using an inverse band-split operation and inverse STFT. By default, the TUSS-DAC cascade is quite inefficient, but we explore recent advances in computational efficiency for both the separation [28] and NAC [13] components.

2.4. SUNAC

To avoid redundant processing, we propose to replace the explicit separation in the cascaded system by a conditional feature extractor in the feature space, leading to the integrated SUNAC architecture illustrated in Fig. 1(c). The encoder maps the input waveform $\mathbf{x}$ into a continuous time–frequency (TF)-like representation $\mathbf{X} \in \mathbb{R}^{F \times T}$, where F and T denote the feature dimension and the number of frames; we adopt a convolutional design following prior work [5, 20]. The conditional feature extractor estimates separated TF-representation based on input learnable prompts, as explained in detail below. The quantizer and decoder are source agnostic, i.e., shared across all sources. The quantizer uses a multi-layer RVQ with projection [5] to discretize the separated TF representations. The decoder then takes the quantized TF features and estimates waveforms $\hat{\mathbf{s}} \in \mathbb{R}^{N \times L}$, where N is the number of prompts provided to the conditional feature extractor. Our decoder combines Transformer [29] and convolutional layers, whereas the SDCodec decoder is convolution-only, thus incurring higher computational cost.

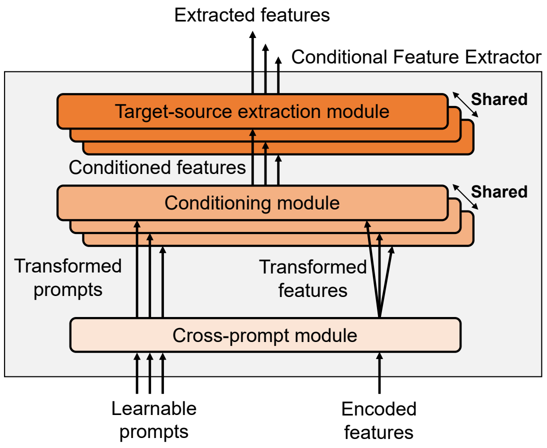

Fig. 2. Detailed architecture of conditional feature extractor

Table 1. Comparison of model parameters and computational cost (MAC) measured per 1.0 second in GMACs. Const denotes the part of the MAC value independent of the number of Sources, while Per source represents the part that scales linearly with the number of Sources. The total MAC for N sources is given by Const + (Per source) $\times N$, where N is the number of sources.

Method	Params (M)	Const [G]	Per source [G]
TUSS	11.1	21.1	10.5
FasTUSS	11.1	4.1	2.1
DAC	74.1		41.0
DACT	66.4		12.9
TUSS-DAC	85.2	21.1	51.5
FasTUSS-DACT	77.5	4.1	14.9
SDCodec	74.8	12.6	28.4
SDCodecT	67.1	3.9	9.0
SUNAC	69.2	3.5	9.5

As shown in Fig. 2, the conditional feature extractor consists of a cross-prompt module, a conditioning module, and a target-source extraction module. The cross-prompt module takes as input the encoded TF representation $\mathbf{X} \in \mathbb{R}^{F \times T}$ and learnable prompt vectors $\mathbf{P} \in \mathbb{R}^{F \times N}$ corresponding to the N prompts $(p_n)_n$. To enable reconstruction of multi-source mixtures, the model is trained to reconstruct the mixture as is when `<Mix>` is supplied. We concatenate the N prompts to $\mathbf{X}$ along the time axis, apply Transformer layers along time, and then split off the first N tokens to yield transformed prompts $\mathbf{P}' \in \mathbb{R}^{F \times N}$ and transformed features $\mathbf{X}' \in \mathbb{R}^{F \times T}$. With positional encoding [30] and self-attention, prompts with the same content but different positions produce different representations. As a result, the input TF features are influenced by the prompts and mapped into a space that facilitates conditional extraction. The conditioning module applies feature-wise linear modulation (FiLM) [31] with residual connection to the transformed features $\mathbf{X}'$ using each transformed prompt $\mathbf{P}'_n \in \mathbb{R}^F$. Using arbitrary trainable functions f and h here implemented as simple linear transformations shared across all prompts, the FiLM output is computed as:

$$\mathrm{FiLM}(\mathbf{X}' | \mathbf{P}'_n) = f(\mathbf{P}'_n) \odot \mathbf{X}' + h(\mathbf{P}'_n), \tag{1}$$

where $\odot$ denotes the element-wise product. We also evaluated

Table 2. Reconstruction results for isolated signals of each source type.

	Speech		Music		SFX	
Model	**SI-SDR ↑**	**VisQOL ↑**	**SI-SDR ↑**	**VisQOL ↑**	**SI-SDR ↑**	**VisQOL ↑**
DAC	8.85 $\pm$ 3.33	4.64 $\pm$ 0.16	7.61 $\pm$ 3.44	4.63 $\pm$ 0.04	2.94 $\pm$ 5.60	4.60 $\pm$ 0.05
DACT	8.42 $\pm$ 3.90	4.79 $\pm$ 0.07	8.66 $\pm$ 4.23	4.57 $\pm$ 0.06	3.16 $\pm$ 5.77	4.54 $\pm$ 0.07
SDCodec[†]	7.24 $\pm$ 4.45	4.63 $\pm$ 0.16	7.65 $\pm$ 4.09	4.56 $\pm$ 0.07	2.24 $\pm$ 6.24	4.54 $\pm$ 0.09
SDCodec	9.11 $\pm$ 3.87	4.67 $\pm$ 0.13	6.63 $\pm$ 4.36	4.23 $\pm$ 0.22	3.36 $\pm$ 5.17	4.55 $\pm$ 0.10
SDCodecT	8.20 $\pm$ 4.29	4.68 $\pm$ 0.12	7.45 $\pm$ 4.57	4.23 $\pm$ 0.22	2.81 $\pm$ 5.67	4.54 $\pm$ 0.07
SUNAC	7.99 $\pm$ 4.39	4.73 $\pm$ 0.11	7.76 $\pm$ 4.11	4.56 $\pm$ 0.06	2.17 $\pm$ 6.63	4.54 $\pm$ 0.07

Table 3. Reconstruction results for the mixture and separated sources from {`<Speech>`, `<Music>`, `<SFX>`} (no repeated prompt).

	Mix		Speech		Music		SFX	
Model	**SI-SDR ↑**	**VisQOL ↑**	**SI-SDR ↑**	**VisQOL ↑**	**SI-SDR ↑**	**VisQOL ↑**	**SI-SDR ↑**	**VisQOL ↑**
TUSS-DAC	–	–	13.07 $\pm$ 2.46	3.72 $\pm$ 0.38	4.70 $\pm$ 4.18	4.25 $\pm$ 0.18	4.82 $\pm$ 5.17	4.19 $\pm$ 0.18
FasTUSS-DACT	–	–	12.29 $\pm$ 2.50	3.53 $\pm$ 0.39	3.92 $\pm$ 4.18	4.26 $\pm$ 0.16	3.80 $\pm$ 5.38	4.17 $\pm$ 0.17
SDCodec[†]	6.39 $\pm$ 3.19	4.52 $\pm$ 0.05	10.78 $\pm$ 2.99	3.50 $\pm$ 0.43	1.74 $\pm$ 3.57	4.26 $\pm$ 0.13	2.17 $\pm$ 4.33	4.20 $\pm$ 0.15
SDCodec	8.24 $\pm$ 2.83	4.56 $\pm$ 0.05	11.40 $\pm$ 3.08	3.64 $\pm$ 0.41	1.21 $\pm$ 3.58	4.10 $\pm$ 0.21	1.26 $\pm$ 4.27	4.18 $\pm$ 0.17
SDCodecT	7.30 $\pm$ 3.06	4.54 $\pm$ 0.06	11.32 $\pm$ 2.89	3.64 $\pm$ 0.37	1.75 $\pm$ 4.45	4.09 $\pm$ 0.22	1.42 $\pm$ 4.76	4.09 $\pm$ 0.22
SUNAC	6.48 $\pm$ 3.11	4.52 $\pm$ 0.06	11.56 $\pm$ 3.00	3.68 $\pm$ 0.36	1.98 $\pm$ 4.62	4.14 $\pm$ 0.20	2.10 $\pm$ 4.94	4.16 $\pm$ 0.19

TUSS-style conditioning via element-wise multiplication with the prompt broadcast over time [24], but that proved less effective in our experiments. Finally, we implement the target-source extraction module with Transformer layers that refine the conditioned TF representation, shared across all prompts.

2.5. Training objective

We can train SUNAC with a permutation-invariant objective [25,26]:

$$\mathcal{L}_{\text{SUNAC}} = \min_{\pi \in \tilde{\mathcal{P}}_S} \sum_{i=1}^{S} \mathcal{L}_{\text{DAC}}\left(s_i, \hat{s}_{\pi(i)}\right) + \mathcal{L}_{\text{DAC}}\left(s_{\text{mix}}, \hat{s}_{\text{mix}}\right), \quad (2)$$

where s_i and $\hat{s}_{\pi(i)}$ are the i-th ground-truth source and its assigned estimate, while s_{mix} and $\hat{s}_{\text{mix}}$ are the ground-truth and estimated mixtures, S is the number of sources, and $\tilde{\mathcal{P}}_S$ is the subset of permutations of $\{1, \dots, S\}$ that only permute indices corresponding to prompts of the same type, leaving others fixed; $\pi \in \tilde{\mathcal{P}}_S$ can thus align, e.g., multiple `<Speech>` estimates to their appropriate references. $\mathcal{L}_{\text{DAC}}$ denotes the DAC loss, which consists of a weighted sum of multi-scale mel-spectrogram loss, adversarial loss, codebook loss, commitment loss, and discriminator loss. The discriminator consists of a multi-period discriminator [32] and a complex multi-scale STFT discriminator [33]. The entire model is trained in a generative adversarial manner, as outlined in [5].

In practice, evaluating all components of the DAC loss over all permutations is computationally prohibitive. Instead, we determine the permutation using a scale-invariant signal-to-distortion ratio (SI-SDR)–based criterion [34] and compute the DAC loss only for the output–reference assignment that minimizes this SI-SDR criterion as follows:

$$\pi^\star = \arg\max_{\pi \in \tilde{\mathcal{P}}_S} \sum_{i=1}^{S} \text{SI-SDR}\big(s_i, \hat{s}_{\pi(i)}\big) \quad (3)$$

$$\mathcal{L}_{\text{SUNAC}} = \sum_{i=1}^{S} \mathcal{L}_{\text{DAC}}\left(s_i, \hat{s}_{\pi^\star(i)}\right) + \mathcal{L}_{\text{DAC}}\left(s_{\text{mix}}, \hat{s}_{\text{mix}}\right) \quad (4)$$

where $\pi^\star$ denotes the optimal permutation of the estimated signals with respect to the reference sources, restricted to permutations among sources of the same type (cross-type indices fixed).

3. EXPERIMENTS

3.1. Experimental conditions

We compare the following methods:
DAC [5]: Single-source reconstruction baseline. The encoder consists of strided downsampling convolutional blocks with factors $(2, 4, 5, 8)$, resulting in a token rate of 50 Hz for 16 kHz audio. Following the encoder, a twelve-layer RVQ module is applied, with each layer using a 1024-entry codebook. The decoder consists of upsampling convolutional blocks that reconstruct the waveform.
DACT: Updated single-source reconstruction model. Relative to DAC, the encoder's convolutional base latent dimension is reduced from 64 to 32, and three Transformer layers (1024 hidden units, 8 attention heads) are added after the convolutional encoder. In the decoder, the base latent dimension is reduced from 1536 to 768, and three Transformer layers (1024 hidden units, 8 heads) are inserted before the convolutional decoder. The resulting architecture is a non-causal version of Mimi [13] that omits semantic tokens.
SDCodec [20]: Extension of DAC separately handling three source types (see Section 2.2). Its encoder and decoder share the same architecture as the 16 kHz version of DAC. Unlike DAC, SDCodec uses three domain-specific RVQs, corresponding to speech, music, and sfx. Each RVQ module is identical to that in DAC.

Table 4. Separated results from {`<Speech>`,`<Speech>`}.

Model	**SI-SDR ↑**	**VisQOL ↑**
TUSS-DAC	13.35 $\pm$ 3.80	4.08 $\pm$ 0.39
FasTUSS-DACT	10.73 $\pm$ 4.66	3.83 $\pm$ 0.46
SDCodec[†]	0.00 $\pm$ 2.83	3.04 $\pm$ 0.61
SDCodec	0.00 $\pm$ 2.83	3.04 $\pm$ 0.62
SDCodecT	0.00 $\pm$ 2.83	3.09 $\pm$ 0.59
SUNAC	11.80 $\pm$ 3.07	4.12 $\pm$ 0.42

Table 5. Reconstruction results for mixed source and separated results from {`<Speech>`, `<Speech>`, `<Music>`, `<SFX>`}.

	Mix		**Speech**		**Music**		**SFX**	
Model	**SI-SDR ↑**	**VisQOL ↑**	**SI-SDR ↑**	**VisQOL ↑**	**SI-SDR ↑**	**VisQOL ↑**	**SI-SDR ↑**	**VisQOL ↑**
TUSS-DAC	–	–	9.07 ± 3.38	3.40 ± 0.47	2.75 ± 3.96	4.20 ± 0.17	3.05 ± 5.23	4.13 ± 0.18
FasTUSS-DACT	–	–	6.98 ± 3.92	3.08 ± 0.38	2.09 ± 3.82	4.21 ± 0.16	2.07 ± 5.37	4.11 ± 0.18
SDCodec†	6.55 ± 2.59	4.49 ± 0.06	-0.95 ± 3.29	2.58 ± 0.53	-0.69 ± 3.64	4.20 ± 0.13	-0.15 ± 4.69	4.15 ± 0.15
SDCodec	8.39 ± 2.31	4.54 ± 0.05	-1.00 ± 3.34	2.64 ± 0.54	-1.62 ± 3.77	4.07 ± 0.21	-0.96 ± 4.44	4.12 ± 0.17
SDCodecT	7.45 ± 2.52	4.51 ± 0.06	-0.95 ± 3.58	2.60 ± 0.56	-1.15 ± 4.45	4.07 ± 0.22	-0.61 ± 4.81	4.11 ± 0.17
SUNAC	6.38 ± 2.54	4.51 ± 0.06	7.46 ± 3.41	3.33 ± 0.45	0.15 ± 4.29	4.11 ± 0.20	0.25 ± 4.97	4.11 ± 0.19

SDCodecT: Updated variant of SDCodec. Relative to SDCodec, both the encoder and the decoder are replaced with that of DACT.
TUSS-DAC: Proposed cascaded system. We employ the TUSS-Medium model for source separation and DAC as the codec, with the two models trained independently. This system is regarded as the upper bound in terms of reconstruction and separation quality.
FasTUSS-DACT: Proposed computationally efficient cascaded system. We use FasTUSS [28] for source separation and DACT as the codec, with the two models trained independently.
SUNAC: Proposed integrated system. The encoder architecture is identical to the convolutional part of the encoder in DACT. The cross-prompt module consists of a single Transformer layer with 1024 hidden units and 8 attention heads, the conditioning module of a FiLM block with a residual connection, and the target source extraction module of two Transformer layers, each with 1024 hidden units and 8 attention heads. Compared with the SDCodec family, SUNAC uses a single RVQ module shared across all domains. The decoder is identical to that of DACT.

We use pre-trained models for DAC [1], TUSS [2], and FasTUSS. For SDCodec, we used two models: (i) the publicly available pretrained model[3], trained with source-count probabilities of 0.6, 0.2, and 0.2 for 1, 2, and 3 sources, respectively; and (ii) an SDCodec separately trained with uniform probability of selecting 1, 2, or 3 sources. The same source-count distribution was also adopted when training the remaining NACs.

For the remaining NACs, including the proposed method, we followed the SDCodec training setup, except that we reduced the batch size to 32 to fit our computational environment.

For SUNAC, we first randomly sample the number of sources $N \in \{1, 2, 3\}$. We then select N sources subject to two constraints: the number of `<Speech>` sources never exceeds two, and `<Music>` and `<SFX>` cannot be repeated in the same mixture.

To evaluate performance, we used the SI-SDR and ViSQOL[4] [35]. Following [20], SI-SDR of the separated codec outputs is computed on signals reconstructed by applying the magnitude mask of each separated source to the input mixture. For ViSQOL, we evaluated the direct output of each decoder. The evaluation was done on the updated Divide and Remaster (DnR) dataset [36], where each speech source contains a single speaker, as well as on a similarly derived dataset where the mixtures additionally include an interfering-speaker source for the two-speaker conditions.

3.2. Experimental results

We compare the number of parameters and the computational cost of each method in Table 1. We quantify computational cost by the number of multiply–accumulate operations (MACs). Replacing the convolutional components with Transformers reduces the computational cost while keeping the parameter count unchanged (e.g., TUSS $\rightarrow$ FasTUSS, DAC $\rightarrow$ DACT, SDCodec $\rightarrow$ SDCodecT). Previous work [28] further showed that, particularly for short audio chunks, most of the compute is spent on convolutions, and that the contributions of the Transformer and convolutional modules remain comparable for 30-s sequences. Therefore, our proposed SUNAC is more computationally efficient than the conventional SDCodec and the cascaded TUSS–DAC, and it remains more efficient than the lighter cascaded version of FasTUSS–DACT. All codecs operate at the same bitrate of 6 kbps. However, the SDCodec family employs source-specific codebooks, whereas the other methods share codebooks across sources.

Hereafter, all values are given as mean ± standard deviation. The symbol † denotes a pretrained model. Table 2 reports reconstruction results for isolated sources, with comparable performance and no meaningful accuracy gaps across methods.

We evaluated the one-output-per-source-type setting with `<Speech>`, `<Music>`, `<SFX>` in Table 3. The results indicate that the proposed SUNAC achieves performance comparable to SDCodec. Compared with cascaded approaches, the SDCodec family and SUNAC tend to achieve lower SI-SDR, which suggests that phase estimation remains challenging. We also note that SI-SDR is computed on decoded signals in Table 2, whereas in Table 3 the SI-SDR for the separated outputs is computed on magnitude-mask estimates, following [20]. However, SUNAC attains VisQOL comparable to FasTUSS-DACT. Table 4 reports two-speaker separation results. Because the SDCodec family cannot handle multiple sources of the same type, its performance is substantially lower, whereas SUNAC achieves results comparable to the cascaded approaches. Table 5 presents the `<Speech>`, `<Speech>`, `<Music>`, `<SFX>` setting, for which TUSS and FasTUSS have been trained but SUNAC has not (we only train SUNAC with up to three prompts). In spite of this handicap, we find SUNAC remains comparable to these cascaded methods.

4. CONCLUSION

We proposed two systems that can generate discrete codes for one or more sources specified by the user from within a mixture: one based on a cascade of separation and NAC, and the other, SUNAC, capable of directly encoding without explicit separation. In both cases, a prompt-based extraction module provides flexible control over the number and type of sources. Experiments show that both systems reliably estimate per-source codes even with multiple sources of the same type, a capability unavailable in conventional source-disentangling codecs such as SDCodec. Furthermore, SUNAC achieves performance comparable to the cascaded pipeline, while offering substantially lower computational complexity. In future work, we will evaluate the learned disentangled representations on relevant downstream tasks.

[1]https://github.com/descriptinc/descript-audio-codec
[2]https://github.com/merlresearch/unified-source-separation
[3]https://github.com/XiaoyuBIE1994/SDCodec
[4]https://github.com/google/visqol

5. REFERENCES

[1] P. Mousavi, G. Maimon, A. Moumen, D. Petermann, J. Shi, H. Wu, H. Yang, A. Kuznetsova, A. Ploujnikov, R. Marxer, B. Ramabhadran, B. Elizalde, L. Lugosch, J. Li, C. Subakan, P. Woodland, M. Kim, H. yi Lee, S. Watanabe, Y. Adi, and M. Ravanelli, "Discrete audio tokens: More than a survey!" *arXiv preprint 2506.10274*, 2025.

[2] P. Mousavi, L. Della Libera, J. Duret, A. Ploujnikov, C. Subakan, and M. Ravanelli, "DASB - discrete audio and speech benchmark," *arXiv preprint arXiv:2406.14294*, 2024.

[3] N. Zeghidour, A. Luebs, A. Omran, J. Skoglund, and M. Tagliasacchi, "SoundStream: An end-to-end neural audio codec," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 30, pp. 495–507, 2021.

[4] A. Défossez, J. Copet, G. Synnaeve, and Y. Adi, "High fidelity neural audio compression," *TMLR*, 2023.

[5] R. Kumar, P. Seetharaman, A. Luebs, I. Kumar, and K. Kumar, "High-fidelity audio compression with improved RVQGAN," in *Proc. NeurIPS*, 2023.

[6] X. Zhang, D. Zhang, S. Li, Y. Zhou, and X. Qiu, "SpeechTokenizer: Unified speech tokenizer for speech large language models," in *Proc. ICLR*, 2024.

[7] J. Casebeer, V. Vale, U. Isik, J.-M. Valin, R. Giri, and A. Krishnaswamy, "Enhancing into the codec: Noise robust speech coding with vector-quantized autoencoders," in *Proc. ICASSP*, 2021.

[8] H. Yang, K. Zhen, S. Beack, and M. Kim, "Source-aware neural speech coding for noisy speech compression," in *Proc. ICASSP*, 2021.

[9] Y. Chae and K. Lee, "Towards bitrate-efficient and noise-robust speech coding with variable bitrate RVQ," in *Proc. Interspeech*, 2025.

[10] X. Luo, J. Huang, R. Yang, Y. Gao, J. Feng, C. Deng, and S. Zhang, "Decodec: Rethinking audio codecs as universal disentangled representation learners," *arXiv preprint 2509.09201*, 2025.

[11] H. Shang, Z. Li, J. Guo, S. Li, Z. Rao, Y. Luo, D. Wei, and H. Yang, "An end-to-end speech summarization using large language model," in *Proc. Interspeech*, 2024.

[12] K. Matsuura, T. Ashihara, T. Moriya, M. Mimura, T. Kano, A. Ogawa, and M. Delcroix, "Sentence-wise speech summarization: Task, datasets, and end-to-end modeling with lm knowledge distillation," in *Proc. Interspeech*, 2024.

[13] A. Défossez, L. Mazaré, M. Orsini, A. Royer, P. Pérez, H. Jégou, E. Grave, and N. Zeghidour, "Moshi: A speech-text foundation model for real-time dialogue," *arXiv preprint arXiv:2410.00037*, 2024.

[14] A. Ohashi, S. Iizuka, J. Jiang, and R. Higashinaka, "Towards a Japanese full-duplex spoken dialogue system," in *Proc. Interspeech*, 2025.

[15] K. Hu, E. Hosseini-Asl, C. Chen, E. Casanova, S. Ghosh, P. Żelasko, Z. Chen, J. Li, J. Balam, and B. Ginsburg, "SALM-Duplex: Efficient and direct duplex modeling for speech-to-speech language model," in *Proc. Interspeech*, 2025.

[16] H. Wang, J. Mao, Z. Guo, J. Wan, H. Liu, and X. Wang, "Leveraging language model capabilities for sound event detection," in *Proc. Interspeech*, 2024.

[17] H. Yin, J. Bai, Y. Xiao, H. Wang, S. Zheng, Y. Chen, R. K. Das, C. Deng, and J. Chen, "Exploring text-queried sound event detection with audio source separation," in *Proc. ICASSP*, 2025.

[18] Y. Zhang, Y. Ikemiya, W. Choi, N. Murata, M. A. Martínez-Ramírez, L. Lin, G. Xia, W.-H. Liao, Y. Mitsufuji, and S. Dixon, "Instruct-MusicGen: Unlocking text-to-music editing for music language models via instruction tuning," in *Proc. ISMIR*, 2025.

[19] Z. Wang, X. Xia, X. Zhu, and L. Xie, "U-SAM: An audio language model for unified speech, audio, and music understanding," in *Proc. Interspeech*, 2025.

[20] X. Bie, X. Liu, and G. Richard, "Learning source disentanglement in neural audio codec," in *Proc. ICASSP*, 2025.

[21] J. Pons, X. Liu, S. Pascual, and J. Serrà, "GASS: Generalizing audio source separation with large-scale data," in *Proc. ICASSP*, 2024.

[22] E. Manilow, G. Wichern, and J. Le Roux, "Hierarchical musical instrument separation," in *Proc. ISMIR*, 2020.

[23] D. Petermann, G. Wichern, A. Subramanian, and J. Le Roux, "Hyperbolic audio source separation," in *Proc. ICASSP*, 2023.

[24] K. Saijo, J. Ebbers, F. G. Germain, G. Wichern, and J. Le Roux, "Task-aware unified source separation," in *Proc. ICASSP*, 2025.

[25] J. R. Hershey, Z. Chen, J. Le Roux, and S. Watanabe, "Deep clustering: Discriminative embeddings for segmentation and separation," in *Proc. ICASSP*, 2016.

[26] M. Kolbæk, D. Yu, Z. H. Tan, and J. Jensen, "Multitalker speech separation with utterance-level permutation invariant training of deep recurrent neural networks," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 25, no. 10, pp. 1901–1913, Jul. 2017.

[27] K. Saijo, G. Wichern, F. G. Germain, Z. Pan, and J. Le Roux, "TF-Locoformer: Transformer with local modeling by convolution for speech separation and enhancement," in *Proc. IWAENC*, 2024.

[28] F. Paissan, G. Wichern, Y. Masuyama, R. Aihara, F. G. Germain, K. Saijo, and J. Le Roux, "FasTUSS: Faster task-aware unified source separation," in *Proc. WASPAA*, 2025.

[29] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. NeurIPS*, 2023.

[30] J. Su, Y. Lu, S. Pan, A. Murtadha, B. Wen, and Y. Liu, "Roformer: Enhanced transformer with rotary position embedding," *arXiv preprint 2104.09864*, 2023.

[31] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. Courville, "FiLM: Visual reasoning with a general conditioning layer," in *Proc. AAAI*, 2018.

[32] J. Kong, J. Kim, and J. Bae, "HiFi-GAN: Generative adversarial networks for efficient and high fidelity speech synthesis," in *Proc. NeurIPS*, 2020.

[33] W. Jang, D. Lim, J. Yoon, B. Kim, and J. Kim, "UnivNet: A neural vocoder with multi-resolution spectrogram discriminators for high-fidelity waveform generation," in *Proc. Interspeech*, 2021.

[34] J. Le Roux, S. Wisdom, H. Erdogan, and J. R. Hershey, "SDR - half-baked or well done?" in *Proc. ICASSP*, 2019.

[35] M. Chinen, F. S. C. Lim, J. Skoglund, N. Gureev, F. O'Gorman, and A. Hines, "ViSQOL v3: An open source production ready objective speech and audio metric," in *Proc. QoMEX*, 2020.

[36] D. Petermann, G. Wichern, Z.-Q. Wang, and J. Le Roux, "The cocktail fork problem: Three-stem audio separation for real-world soundtracks," in *Proc. ICASSP*, 2022.