

ESPNET-CODEC: COMPREHENSIVE TRAINING AND EVALUATION OF NEURAL CODECS FOR AUDIO, MUSIC, AND SPEECH

Jiatong Shi^{1*}, Jinchuan Tian^{1*}, Yihan Wu^{1,2*}, Jee-weon Jung^{1†}, Jia Qi Yip^{3†}, Yoshiki Masuyama^{4†}, William Chen^{1†}, Yuning Wu^{2†}, Yuxun Tang^{2†}, Massa Baali¹, Dareen Alharthi¹, Dong Zhang⁶, Ruifan Deng⁶, Tejes Srivastava⁷, Haibin Wu⁷, Alexander Liu⁹, Bhiksha Raj¹, Qin Jin², Ruihua Song², Shinji Watanabe¹

¹ Carnegie Mellon University, ² Renmin University of China,

³ Nanyang Technological University, ⁴ Tokyo Metropolitan University, ⁵ Fudan University,

⁶ University of Chicago, ⁷ National Taiwan University, ⁸ Massachusetts Institute of Technology

ABSTRACT

Neural codecs have become crucial to recent speech and audio generation research. In addition to signal compression capabilities, discrete codecs have also been found to enhance downstream training efficiency and compatibility with autoregressive language models. However, as extensive downstream applications are investigated, challenges have arisen in ensuring fair comparisons across diverse applications. To address these issues, we present a new open-source platform ESPnet-Codec, which is built on ESPnet and focuses on neural codec training and evaluation. ESPnet-Codec offers various recipes in audio, music, and speech for training and evaluation using several widely adopted codec models. Together with ESPnet-Codec, we present VERSA, a standalone evaluation toolkit, which provides a comprehensive evaluation of codec performance over 20 audio evaluation metrics. Notably, we demonstrate that ESPnet-Codec can be integrated into six ESPnet tasks, supporting diverse applications.

Index Terms— Neural codecs, codec evaluation

1. INTRODUCTION

Speech representation, derived from speech signals, is fundamental for various speech tasks, including automatic speech recognition (ASR) and text-to-speech (TTS). Recently, self-supervised learning (SSL) has emerged as a promising method in speech representation learning, leveraging unlabeled data to learn useful representations that surpass previous state-of-the-art results on various benchmarks [1–6]. However, continuous SSL representations in downstream tasks often have scalability issues related to storage and computation due to their higher dimensionalities (e.g., 1, 024 of WavLM-Large) compared to conventional acoustic features (e.g., 48 of mel spectrogram) [7–10]. This has inspired a growing interest in learning discrete speech representation that aims to offer more efficient and compact representations for downstream tasks [9, 11–21].

One of the main streams in discrete speech representation is the neural codec type of approach. Initially proposed for audio compression, neural codecs are usually based on an encoder-decoder architecture, equipped with a quantizer that transmits intermediate representations to discrete codes [22–24]. While the original focus of neural codecs was signal compression, the interest in applying discrete representations for speech-related tasks has extended their usage into more downstream speech and audio tasks [16–20].

Although neural codecs have become the basis for many downstream applications, several difficulties remain in comparing neural codecs in a controlled setup. These difficulties include, but are not

limited to, variations in training data, training environment, and evaluation settings. Such challenges further complicate the evaluation of downstream applications, potentially impacting the corresponding scientific findings. To ensure a reasonable analysis of experimental findings, a framework that integrates various downstream tasks would be highly beneficial for future studies in both speech discrete representations and their downstream applications.

In this work, we propose a new toolkit, ESPnet-Codec, built on the ESPnet platform [25]. ESPnet-Codec currently supports five neural codec algorithms and will be further expanded. The toolkit integrates seamlessly with existing ESPnet downstream tasks. Following the design principles of previous ESPnet toolkits [25–32], ESPnet-Codec includes a core PyTorch-based library and a collection of recipes in audio, music, and speech. These recipes follow an all-in-one design, encompassing data preparation, model training, model inference, and evaluation.¹

In addition to releasing ESPnet-Codec, this paper also provides an extensive comparative study of neural codecs and their applications. We propose another standalone toolkit named VERSA (Versatile Speech and Audio Evaluation toolkit), which is able to evaluate a wide array of existing codec algorithms for speech/audio/music generation tasks. With VERSA, we offer a comprehensive analysis within a highly controlled experimental environment. While demonstrating that ESPnet-Codec has comparable performance to existing toolkits, our experiments also show that no single model can achieve the best performance across all metrics, suggesting the necessity for comprehensive evaluation toolkits like VERSA. Furthermore, a major benefit of ESPnet-Codec is its tight integration with various downstream tasks. We test pre-trained codecs across six downstream tasks and demonstrate how codec performance correlates with effectiveness in these tasks. While previous codec-based applications have mostly focused on generation tasks, we demonstrate feasible integration with understanding tasks and highlight the limitations of existing codec models.

2. RELATED WORKS

While existing open-source works in neural codecs have made significant contributions to the community [17, 18, 23, 24, 33–36], they still face issues in codec training and integration with other tasks, as detailed in Table 1. First, although most toolkits provide codec training scripts, they usually do not provide all-in-one recipes with data preparation, training, inference, evaluation, and pre-trained model weights available. The limitation then complicates the comparison

¹All codebase, data preprocessing scripts, and experimental configurations are released at [ESPnet](#). Pre-trained models are released at [Huggingface](#).

*co-first authors. †co-second authors.

Table 1: A comparison of existing codec-related toolkits. Codec types refer to the framework of different codec models, while recipes indicate whether the toolkit provides dataset-related training, inference, and evaluation pipelines. (The information is collected in June, 2024.)

Toolkit	Release	# Codecs	Open-source Level					# Recipes	Supported Downstream Tasks using Codec							
			Data	Training	Inference	Evaluation	Weights		ASR	TTS	TTM	TTA	SSE	SVS	SPK	SSL
Encodec [23]	2022/10	1	✗	✗	✓	✗	✓	0	✗	✗	✗	✗	✗	✗	✗	✗
AudioDec [33]	2023/05	1	✓	✓	✓	✗	✓	0	✗	✗	✗	✗	✓	✗	✗	✗
AcademiCodec [34]	2023/05	3	✗	✓	✓	✓	✓	0	✗	✗	✗	✗	✗	✗	✗	✗
DAC [24]	2023/06	2	✗	✓	✓	✓	✓	0	✗	✗	✗	✗	✗	✗	✗	✗
AudioCraft [17]	2023/06	1	✓	✗	✓	✓	✗	0	✗	✗	✓	✓	✗	✗	✗	✗
SpeechTokenizer [18]	2023/08	1	✗	✓	✓	✗	✓	0	✗	✓	✗	✗	✗	✗	✗	✗
FunCodec [35]	2023/09	3	✓	✓	✓	✓	✓	1	✗	✓	✗	✓	✗	✗	✗	✗
Amphion [36]	2023/12	1	✗	✗	✓	✗	✓	0	✗	✓	✗	✗	✗	✗	✗	✗
ESPnet-Codec	2024/06	5	✓	✓	✓	✓	✓	7	✓	✓	✗	✗	✓	✓	✓	✓

of models from different toolkits [17, 18, 23, 24, 34, 36]. Notably, some of the best publicly available pre-trained models are not associated with public datasets [23, 35].

Moreover, there has been a need for a unified yet comprehensive evaluation toolkit for signal quality measures. Existing toolkits either do not support in-toolkit evaluation or provide limited evaluation metrics. In Table 1, AcademiCodec, DAC, AudioCraft, and FunCodec provide evaluation protocols, but all of them support only a few metrics for evaluation.² When using only one or two metrics, potential biases can be easily introduced (see Sec. 4 for analysis). In contrast, using a wider range of metrics allows for a more comprehensive analysis and understanding of different codec models.

Additionally, current frameworks’ integrations of downstream tasks only focus on limited generation tasks. However, considering the growing needs in spoken language modeling and the diverse usage of codec tokens [9, 12–15, 20], incorporating diverse tasks can benefit not only direct applications but also our understanding of different codec algorithms in views from downstream applications (see Sec. 4 for some analysis).

To address these limitations, we propose ESPnet-Codec, built on the foundation of existing ESPnet tasks, including ASR, TTS, speaker recognition (SPK), speech separation&enhancement (SSE), singing voice synthesis (SVS), and self-supervised learning pre-training (SSL) [25–32]. With ESPnet-style recipes, this toolkit provides transparent comparisons among codec algorithms by re-implementation within the ESPnet ecosystem. The codec models can be thoroughly evaluated using the latest speech evaluation framework by VERSA. Additionally, the model supports integration with six existing ESPnet tasks through a discretization interface.

3. FUNCTIONALITIES

3.1. General Neural Codec Framework

ESPnet-Codec follows general neural codec frameworks: The source input for the codec model is a sampled audio signal $S \in \mathbb{R}^{1 \times T_s}$ with a length of T_s samples.³ The codec model M has an Encoder(\cdot), a quantizer Quantizer(\cdot), and a decoder Decoder(\cdot). In the quantizer, we define a number of L codebooks (i.e., $\{\mathcal{B}_1, \dots, \mathcal{B}_L\}$) where the i^{th} codebook $\mathcal{B}_i = \{1, 2, \dots, B_i\}$ has B_i codes. The forward process of codec M is a typical serial pipeline. The encoder first converts S into hidden states $E \in \mathbb{R}^{D \times T_e}$ with a sequence length of T_e and a

dimension of D (i.e., $E = \text{Encoder}(S)$). Based on E , the quantizer generates discrete codes $C \in (\mathcal{B}_1, \mathcal{B}_2, \dots, \mathcal{B}_L)^{T_e}$ with T_e discretized frames, where C can be interpreted into $\hat{E} \in \mathbb{R}^{D \times T_e}$ with the codebooks $\{\mathcal{B}_1, \dots, \mathcal{B}_L\}$ (i.e., $(C, \hat{E}) = \text{Quantizer}(E|\mathcal{B}_1, \mathcal{B}_2, \dots, \mathcal{B}_L)$). Lastly, the decoder utilizes \hat{E} to obtain \hat{S} , aiming to reconstruct the original signal S (i.e., $\hat{S} = \text{Decoder}(\hat{E})$).

Sound Representation. ESPnet-Codec supports the use of both raw waveform and spectrogram from short-time Fourier transform (STFT). The spectral option provides flexibility in supporting codecs based on spectral properties or complex domain [35, 41].

Encoder and Decoder Architecture. Due to the efficiency concern, existing codec algorithms mostly use convolutional encoders and decoders for pre/post quantization processing. Follow existing codec implementations [17, 23], ESPnet-Codec supports SEANet encoder and decoder [42] and their variants in [18, 24, 33, 34]. For additional decoder types, as suggested in [33], ESPnet-Codec also supports non-symmetric decoders following the vocoder designs, sourced from ESPnet2-TTS [27].

Quantization Algorithm. ESPnet-Codec primarily supports residual vector quantization (RVQ) as the quantization algorithm, where each VQ follows the VQ-variational auto-encoder (VQ-VAE)-based optimization with exponential moving average [43]. A few variants of RVQ, including Group-RVQ (GRVQ) proposed in [34], are also supported.

Training and Loss Functions. We primarily select the generative adversarial network (GAN)-based training framework for the codec trainer. Aligned with existing toolkits [17, 35], ESPnet-Codec has training objectives in three focuses: reconstruction, adversarial, and quantization focuses.

For reconstruction losses, ESPnet-Codec offers time-domain speech signal loss and frequency-domain mel spectrogram losses:

$$\mathcal{L}_{\text{rec}}(S, \hat{S}) = \|S - \hat{S}\|_{\text{norm}} + \frac{1}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} \|M_a(S) - M_a(\hat{S})\|_{\text{norm}}, \quad (1)$$

where $\|\cdot\|_{\text{norm}}$ can be either L1 norm, L2 norm, or their combination; \mathcal{A} is a set of scales for different mel spectrogram;⁴ M_a is the corresponding mel spectrogram extractor to the scale a . Additionally, we support distillation losses from teacher-student learning, which utilize pre-trained SSL models as in [18].

For adversarial losses, the generator has both the discriminator losses and a feature matching loss:

$$\mathcal{L}_{\text{gen}} = \frac{1}{K} \sum_k \max(0, 1 - D_k(\hat{S})) + \frac{1}{K \cdot R} \sum_{k,r} \|D_k^r(S) - D_k^r(\hat{S})\|_1, \quad (2)$$

⁴The scale here refers to the STFT parameters. Default in our models, we use window sizes of $\{2^5, 2^6, \dots, 2^{11}\}$ with the frame shifts of $\frac{1}{4}$ window sizes.

²AcademiCodec: PESQ [37] and STOI [38]; DAC: training losses, SI-SDR [39], and VISQOL [40]; AudioCraft: VISQOL and SI-SNR [39]; Fun-Codec: VISQOL and word error rates from ASR systems.

³Depending on the codec model, preprocessing such as transformations to the frequency domain can be conducted for following codec modules.

where D_k is the k^{th} discriminator, r represents the r^{th} layer, K is the total number of discriminator, and R is the number of outputs in the discriminator. Discriminators are optimized with:

$$\mathcal{L}_{\text{disc.}} = \frac{1}{K} \sum_k [\max(0, 1 + D_k(\hat{S})) + \max(0, 1 - D_k(S))]. \quad (3)$$

ESPnet-Codec supports up to six variants of discriminators. Inherited from ESPnet2-TTS [27], we support the multi-resolution STFT discriminator from ParallelWaveGAN [44], the filter-bank random window discriminator (FB-RWD) from StyleMelGAN [45], the multi-scale discriminator (MSD) and the multi-period discriminator (MPD) from HiFi-GAN [46], the simple STFT-based discriminator from SoundStream [22], the multi-scale complex STFT discriminator from Encodec [23], and the multi-scale multi-period multi-band (MSMPMB) discriminator from DAC [24].

For quantization losses, we consider commitment losses from both the whole quantizer and different levels of the quantizer as in:

$$\mathcal{L}_{\text{quan.}} = \|E - (\hat{E})\|_1 + \frac{1}{L} \sum_{i=1}^L \|Q_{i-1} - VQ_i(Q_{i-1})\|_{\text{norm}}, \quad (4)$$

where Q_i is the i^{th} hidden states to be encoded and Q_0 is identical to E . L , as defined above, is the number of codebooks (i.e., the level of the RVQ quantizer). VQ_i is the i^{th} quantizer.

3.2. Example Neural Codec Models

SoundStream: SoundStream [22] represents the initial effort in applying RVQ to neural codec learning. While the generator has a SEANet-based encoder-decoder and the RVQ quantizer, the model has two discriminators consisting of a waveform-based discriminator and a spectral discriminator. The losses in SoundStream include adversarial losses and feature-matching losses from the discriminators, as well as a multi-resolution mel loss.

Encodec: Encodec [23] follows a similar architecture to SoundStream but uses a multi-scale STFT discriminator to replace the STFT discriminator. During training, the discriminator is not updated at p_{skip} probability. Furthermore, a loss balancer is used to automatically synchronize the loss scales.

DAC: DAC [24] extends Encodec with a few updates, including the utilization of the snake activation function, improved RVQ with factorized codes and L2-normalized codes, increased quantizer dropout, and the use of an MSMPMB discriminator.

FunCodec: While FunCodec is proposed as a toolkit [35], it introduces the interface of using complex STFT to train codec in the frequency domain. The codec model named FunCodec in ESPnet follows the “FreqCodec” configuration in the FunCodec repository, which utilizes a multi-scale STFT discriminator.

HiFi-Codec: HiFi-Codec [34] improves on Encodec by using GRVQ to split the hidden states into multiple sub-groups, where each sub-group applies RVQ for reconstruction independently. Moreover, HiFi-Codec employs three discriminators, including MSD, MPD, and multi-scale STFT discriminators.

3.3. Speech and Audio Quality Evaluation with VERSA

As discussed in Sec. 2, previous codec-related toolkits typically support a limited set of evaluation metrics, leading to discrepancies in metric selection. Moreover, reported numbers can vary due to differences in hyper-parameter setups or the downstream models used for evaluation. Additionally, some metrics, while demonstrating superior quality in the contexts for which they were designed, may not be as robust under diverse conditions (see Section 4 for details). Relying on a limited set of metrics is risky for codec model selection,

Table 2: Evaluation metrics supported in the VERSA toolkit. Evaluation metrics are listed in abbreviations. Details are in Sec. 3.3.

Dependency	Base	Evaluation Metrics
Intrusive	Non-Learning	MCD, F0-RMSE, F0-CORR, SI-SNR, CI-SDR, PESQ, STOI
	Learning	VISQOL, D-BLEU, D-Distance, S-BERT
Non-intrusive		DNSMOS, UTMOS, PLCMOS, SingMOS
Other Perceptual		CER/WER, SPK-SIM

especially when codec models often serve as fundamental modules for other systems. To address these issues, we introduce a standalone Versatile Speech and Audio Evaluation toolkit (VERSA) alongside ESPnet-Codec to unify the evaluation framework. VERSA supports both intrusive (full-reference), non-intrusive (no-reference) metrics, and other perceptual metrics that are not directly related to signal-level information. The intrusive category includes both non-learning and learning-based measures, while the non-intrusive category and other perceptual metrics consist solely of learning-based measures.

For intrusive metrics, we support non-learning measures, including mel cepstral distortion (MCD), F0 root mean square error (F0-RMSE), F0 Pearson correlation (F0-CORR), scale-invariant signal-to-noise ratio (SI-SNR) [39], convolutive transfer function invariant signal-to-distortion ratio (CI-SDR) [47], perceptual evaluation of speech quality (PESQ) [37], and short-time objective intelligibility (STOI) [38]. For learning-based metrics, VERSA supports virtual speech quality objective listener (VISQOL) [40], speech BLEU score (D-BLEU) [48], discrete speech distance (D-Distance) [48], and speech BERT score (S-BERT) [48]. For non-intrusive metrics, we support deep noise suppression mean opinion score (DNS-MOS) [49], UTokyo-SaruLab system for VoiceMOS Challenge (UT-MOS) [50], packet loss concealment MOS (PLCMOS) [51], and singing MOS (SingMOS) [52]. For other perceptual metrics, VERSA includes character/word error rate (CER/WER) by pre-trained speech recognition models from either ESPnet or OpenAI-Whisper [25, 53], Speaker similarity (SPK-SIM) powered by more than ten speaker-embedding extractors based on ESPnet-SPK [29]. By providing this comprehensive suite of evaluation tools, VERSA aims to standardize and streamline the assessment of codec performance across different models and applications.

3.4. Downstream Application

This section showcases several downstream applications of pre-trained ESPnet-Codec models. While we demonstrate the ease and practicality of applying the codec to various downstream tasks, these tasks also assess the generalizability, robustness, and efficiency of different speech neural codecs.

ASR transcribes text from speech. Recent studies show that ASR based on discrete representation (i.e., discrete ASR) can achieve efficient training, reduced storage constraints, and improved performance over spectral features [8, 9, 15, 21, 54–57]. While most previous works focus on discrete units from pre-trained SSL models, existing discrete ASR systems using neural codecs are typically presented in a multi-task manner with large-scale training data [54, 58].

Supported by ESPnet [9], the framework for discrete ASR, pre-trained Codec models can be easily integrated into the discrete ASR task as model input. The discrete tokens from neural codecs can either be mapped to embeddings or looked up from the existing codebooks in the codec models. Embeddings from multi-level codebooks are then fed to the downstream ASR model for ASR training, enjoying the full capacity of ESPnet ASR architectures.

TTS has been a major application for neural codecs. Before

the utilization of discrete representation, TTS was dominated by non-autoregressive (NAR) algorithms [59, 60] due to their efficiency, controllability, and stable performance over their autoregressive (AR) counterparts [61]. However, the introduction of discrete neural codecs has brought additional attention to AR modeling [16, 19, 62, 63], which has proven versatile in generating expressive speech with robustness in zero-shot scenarios. Meanwhile, recent works have shown that NAR TTS can also benefit from using pre-trained codecs in their modeling [64, 65]. In this work, on top of foundations from ESPnet-TTS [26, 27] we develop two discrete TTS tasks with ESPnet-Codec to support both NAR and AR TTS.

NAR TTS. For NAR TTS, we follow the discrete TTS approach proposed in [57]. The model employs a FastSpeech2-like backbone as in [21] with a variational auto-encoder-based approach to achieve NAR TTS without duration supervision. The prediction targets for the NAR TTS are codecs from reference speech signals. Multi-speaker TTS is achieved by using pre-trained speaker embeddings.

AR TTS (Speech LM). For AR TTS, we implement VALL-E [16]. The model uses an AR decoder-only model to predict the first stream of codecs from the given text and a target speaker speech prompt. Then, the following streams are predicted in a NAR fashion based on the first codec stream.

SPK (i.e., the SPK task in the paper) focuses on identifying the speaker from speech signals. While existing SPK models primarily depend on spectral features, raw waveform, or SSL features [2, 66, 67], recent work has shown the potential of discrete neural codecs as an alternative to existing speech features [68].

As an important attribute of speech signals, ESPnet-Codec can also be adopted for the SPK task by integrating with ESPnet-SPK [29], simply by replacing the acoustic input with codecs.

SSE aims at extracting the target speech from a mixture of overlapping speakers with background noise. Typical approaches to SSE have leveraged STFT [69, 70], a trainable encoder [71, 72], and a pre-trained SSL [73] to obtain redundant representation. Although a recent work [74] has shown the potential SSE on neural codecs, compressed representations tend to cause degraded performance on objective metrics [75]. Hence, a renewed emphasis on perceptual evaluation is required to clarify the advantages of codecs.

ESPnet-Codec supports SSE on top of ESPnet-SE [74] and implements additional flexibility in switching between activation functions, masking, and mapping approach, which have been identified as factors impacting the performance of codec-based SSE.

SVS targets the synthesis of singing voices from musical inputs. Compared to the similar task of TTS, SVS has strict requirements for rhythm control and a greater sensitivity to high-frequency information in the signal. The concept of discrete SVS was introduced in the discrete speech challenge at Interspeech 2024 [21], prompting several investigations into the application of pre-trained SSL tokens to SVS [76, 77]. The winning system, “TokSing,” demonstrated superior performance and efficiency compared to models using spectral features or VAE-based latent features [76, 78, 79].

Building on Muskits [28], we implement the TokSing architecture within the context of neural codecs using ESPnet-Codec. Similar to TokSing, we predict multi-stream neural codecs in parallel, serving as the training targets for the SVS task.

SSL in speech/audio signals has proven to be an effective way of utilizing unlabeled data, especially in extracting semantic features [1–3, 32]. Existing speech/audio SSL models mostly rely on the raw waveform, which can incur a large computational overhead in the feature extractor [80–83].

While existing methods focus on using spectral features to replace the convolutional feature extractor [80–83], ESPnet-Codec

connects to the ESPnet-SSL task [32] to support Codec-based SSL for the first time, replacing the raw waveform with discrete codes.

4. EXPERIMENTS

4.1. Codec Dataset

LibriTTS (base dataset): Aligned with prior studies, we exemplify the toolkit on the LibriTTS dataset [84]. Specifically, we use the 460-hour training set for codec training and the test-clean set for evaluation. To study the effect of different sampling rates, we evaluate the model at both 16kHz and 24kHz sampling rates.

AMUSE (large-scale dataset): Additionally, we include all the datasets used in DAC [24] to gather a range of open-source high-quality datasets, forming a new fused dataset named the Audio, Music, and Speech Ensemble (AMUSE) corpus. All datasets in AMUSE have a sampling rate of 44.1kHz or higher. In addition to the datasets used in the DAC training set, AMUSE includes speech training data from AISHELL3 [85], Google18n-TTS corpora, and Mexican endangered languages [86, 87].⁵ For music training data, AMUSE incorporates additional singing voice data from OpenSinger, StyleSing111, M4Singer, Kiritan-singing, Oniku Kurumi Utagee database, Natsume Singing database, Opencpop, ACE-KiSing (excluding original voices), PJS, and JSUT singing [88–95]. We randomly sample 1,000 utterances from the training set to form the development set. For the test set, we collect the test sets of all datasets that explicitly include test sets. Given that the singing voice datasets are relatively small, we add a few more datasets to the test set, including PopCS, Ofuton P Utagee database, and ACE-KiSing-original as additional test sets [93, 96]. We use the full music test set. Since the test set is relatively large, we subsample the test set to 3,000 utterances for audio and speech, respectively, using a fixed random seed. For AMUSE-related experiments, we investigate both 16kHz and 44.1kHz sampling rates.⁶

4.2. Codec Experimental Setup

We conduct experiments with codecs in a controlled setting, including a base setting on LibriTTS and a large-scale setting on AMUSE. For the base setting, we train models using a single V100 GPU, a batch size of 8 with 1-second chunks, and 600k training steps. For the large-scale setting, models are trained using four Ampere GPUs, a batch size of 128 with 2-second chunks, and 600k training steps. Since all supported codecs use RVQ as their VQ strategy, we uniformly use 32-level codebooks, each with 1,024 codes. The training adopts multi-band learning with random bitrate control of 2, 4, 8, 16 kbps. The weights of losses for each model are tuned based on development sets. For codec configurations, we mostly follow the settings in their original papers or publicly released configurations [18, 22–24, 34, 35]. A few specific changes are made based on hyperparameter tuning on the development sets.⁷ For comparison,

⁵As mentioned in [24], its original collection of speech data has diverse qualities because of upsampling from low-sampling rate signals. Therefore, we intentionally add more TTS-focused data with known high quality into AMUSE to balance the training.

⁶A few recipes related to AMUSE will be released with ESPnet-Codec, which additionally provides codec options for {audio, music, speech} only scenarios for related usages. ^aBecause of the space limitation, we do not present their results in the paper.

⁷For Encodec, we do not use the loss balancer as it did not show effectiveness on the development set. For DAC, we do not include factorization in the code space, as we empirically find the low-dimensional hidden states difficult to converge in our experiments.

Table 3: LibriTTS codec performance comparison at {16 / 24}kHz. * indicates models trained using existing open-source codebases.

Model	Intrusive										Non-Intrusive				Perceptual	
	Non-learning						Learning									
	MCD ↓	F0-RMSE ↓	F0-CORR ↑	STOI ↑	PESQ ↑	CI-SDR ↑	D-BLEU ↑	D-Distance ↑	S-BERT ↑	VISQOL ↑	UTMOS ↑	PLCMOS ↑	DNSMOS ↑	WER ↓	SPK-SIM ↑	
DAC* [24]	13.85 / 3.17	61.03 / 52.09	0.61 / 0.72	0.98 / 0.99	3.81 / 3.45	15.48 / 14.02	0.88 / 0.93	0.91 / 0.94	0.98 / 0.99	2.90 / 4.60	3.81 / 3.95	4.01 / 4.19	3.05 / 3.02	2.9 / 2.7	0.85 / 0.97	
Encodec* [23]	4.84 / 3.94	42.04 / 38.00	0.59 / 0.65	0.97 / 0.98	3.45 / 4.09	9.89 / 11.38	0.85 / 0.89	0.90 / 0.92	0.98 / 0.99	4.49 / 4.73	3.86 / 4.08	4.16 / 4.27	3.06 / 3.09	2.4 / 2.4	0.86 / 0.95	
FunCodec* [35]	5.40 / 5.52	43.19 / 43.81	0.55 / 0.55	0.90 / 0.93	3.23 / 3.11	11.46 / 36.29	0.78 / 0.79	0.87 / 0.87	0.95 / 0.96	4.57 / 4.52	3.84 / 3.84	4.48 / 4.22	3.15 / 3.14	2.2 / 2.1	0.93 / 0.92	
SpeechTokenizer* [18]	5.46 / -	43.32 / -	0.55 / -	0.93 / -	2.62 / -	72.63 / -	0.80 / -	0.87 / -	0.95 / -	4.36 / -	3.80 / -	4.08 / -	3.06 / -	2.2 / -	0.62 / -	
SoundStream	4.64 / 4.28	43.02 / 41.08	0.57 / 0.57	0.95 / 0.93	2.83 / 2.54	50.75 / 93.19	0.83 / 0.78	0.89 / 0.86	0.96 / 0.94	4.45 / 4.21	3.76 / 3.58	4.38 / 4.03	3.10 / 2.97	2.1 / 2.1	0.84 / 0.76	
Encodec	3.73 / 4.26	43.57 / 46.12	0.55 / 0.53	0.97 / 0.97	3.37 / 3.25	51.31 / 92.68	0.87 / 0.87	0.91 / 0.91	0.98 / 0.97	4.65 / 4.55	3.92 / 3.93	4.45 / 4.17	3.11 / 3.12	2.1 / 2.1	0.90 / 0.86	
DAC	5.16 / 4.98	65.20 / 62.92	0.56 / 0.53	0.95 / 0.97	3.00 / 3.43	48.84 / 93.51	0.85 / 0.87	0.89 / 0.90	0.97 / 0.97	4.48 / 4.21	3.78 / 3.93	4.28 / 4.07	3.04 / 3.05	2.9 / 2.8	0.76 / 0.83	
HiFi-Codec	6.03 / 6.37	42.33 / 42.58	0.56 / 0.56	0.94 / 0.95	2.48 / 2.42	71.33 / 96.81	0.78 / 0.77	0.86 / 0.85	0.94 / 0.93	4.01 / 3.96	3.61 / 3.58	4.23 / 3.88	2.89 / 2.89	3.1 / 3.7	0.66 / 0.65	
Fun-Codec	5.87 / 5.76	67.72 / 69.90	0.54 / 0.51	0.83 / 0.86	2.43 / 2.40	50.23 / 89.73	0.71 / 0.73	0.82 / 0.83	0.90 / 0.91	4.26 / 4.24	3.01 / 3.09	4.13 / 3.90	3.04 / 3.03	3.7 / 3.6	0.72 / 0.70	
Ground Truth	-	-	-	-	-	-	-	-	-	-	4.05	4.42	3.08	2.1	-	

Table 4: AMUSE codec performance comparison on the *speech* test set at {16 / 44.1}kHz.

Model	Intrusive										Non-Intrusive		Perceptual	
	Non-learning					Learning								
	MCD ↓	F0-RMSE ↓	F0-CORR ↑	STOI ↑	PESQ ↑	CI-SDR ↑	D-BLEU ↑	D-Distance ↑	S-BERT ↑	VISQOL ↑	UTMOS ↑	PLCMOS ↑	DNSMOS ↑	SPK-SIM ↑
SoundStream	4.77 / 5.87	42.95 / 46.67	0.51 / 0.49	0.92 / 0.94	2.71 / 2.79	63.12 / 60.64	0.71 / 0.73	0.88 / 0.88	0.95 / 0.96	4.51 / 4.49	1.95 / 1.93	3.78 / 3.65	2.66 / 2.59	0.87 / 0.82
Encodec	4.91 / 5.14	46.73 / 57.64	0.46 / 0.37	0.91 / 0.91	2.28 / 1.81	60.57 / 58.57	0.70 / 0.73	0.87 / 0.88	0.94 / 0.95	4.52 / 4.45	1.55 / 1.59	3.64 / 2.17	2.54 / 2.44	0.85 / 0.85
DAC	5.76 / 4.99	44.36 / 43.19	0.51 / 0.55	0.91 / 0.97	2.75 / 4.02	66.08 / 60.20	0.68 / 0.79	0.86 / 0.90	0.94 / 0.98	4.29 / 4.67	2.14 / 2.37	3.65 / 2.34	2.77 / 2.84	0.78 / 0.93
Ground Truth	-	-	-	-	-	-	-	-	-	-	2.45	3.85	2.79	-

Table 5: AMUSE codec performance comparison on the *audio* test set at {16 / 44.1}kHz.

Model	MCD ↓	CI-SDR ↑	VISQOL ↑
SoundStream	3.57 / 5.05	2.47 / 82.01	4.38 / 3.92
Encodec	3.59 / 4.90	-4.64 / 73.14	4.32 / 4.03
DAC	3.59 / 4.97	7.72 / 75.84	4.23 / 3.90

Table 6: AMUSE codec performance comparison on the *music* test set at {16 / 44.1}kHz.

Model	MCD ↓	CI-SDR ↑	VISQOL ↑
SoundStream	3.86 / 5.60	5.41 / 70.51	3.77 / 3.93
Encodec	4.20 / 5.60	0.00 / 65.79	3.81 / 3.81
DAC	3.66 / 5.14	14.61 / 65.49	3.52 / 3.67

we also train the codec models on LibriTTS based on the original DAC, Encodec, FunCodec, and SpeechTokenizer [17, 18, 24, 35].

For evaluation, resampling is conducted for metrics that are fixed to a specific sampling rate. The layer index and KMeans models for discrete-speech evaluation are based on recommended values from [48]. We use the speech version of VISQOL for speech evaluation, but the default version for audio and music evaluation [40]. The CER is calculated based on Whisper-large-V3 for codecs trained on LibriTTS, and the SPK-SIM is based on cosine similarity from pre-trained RawNet3 [66] speaker embeddings from ESPnet-SPK [29].

4.3. Codec Results and Discussion

The results of LibriTTS-base experiments are shown in Table 3. In general, our reproduced codec models achieve performance comparable to existing codec implementations, with notably better performance on CI-SDR and WER by Whisper. We also observe diverse differences across various evaluation metrics, as no single model achieves the best scores in all metrics. For instance, while the original DAC performs better in several metrics, it has much worse WER by Whisper compared to FunCodec. This finding supports our motivation for proposing VERSA, as discussed in Sec. 2. Within ESPnet-Codec models, Encodec shows the best performance in most metrics, while other models have their own strengths, such as SoundStream’s superior F0 tracking and DAC’s better PESQ in 24kHz speech.

Table 4, 5, and 6 show the evaluation results on AMUSE speech, audio, and music test sets, respectively. It is clear that large-scale data significantly impacts all three evaluated codec models, completely altering their relative performance on each metric. Notably, while Encodec performs well on LibriTTS, it does not generalize as well to large-scale data in speech (Table 4). Similar to the base experiments on LibriTTS, there is no agreement on all the evaluation metrics. In Table 4, Encodec has worse scores in three MOS

Table 7: CodecSUPERB benchmark performance (hidden set).+ indicate models trained on large-scale AMUSE dataset.

Model	Application				Signal Metrics			
	ASR		SPK		Speech		Audio	
	WER ↓	EER	ACC ↑	ACC ↑	PESQ ↑	STOI ↑	Mel Loss ↓	Mel Loss ↓
SoundStream	5.99	2.80	49.49	51.13	2.54	0.91	0.87	2.21
Encodec	5.58	2.20	52.53	61.58	2.99	0.95	0.74	1.02
DAC	6.33	3.40	55.56	52.65	2.51	0.91	0.90	1.25
SoundStream+	5.88	2.40	53.54	68.11	2.95	0.93	0.74	0.96
Encodec+	5.98	2.40	50.51	66.50	2.48	0.92	0.80	0.97
DAC+	6.08	3.20	54.55	61.89	2.94	0.93	0.86	1.08
Original Audio	5.28	1.60	59.60	78.01	-	-	-	-

scores but still has comparable VISQOL scores to other codec models. Similarly in Table 5 and 6, we observe a mixed result over different settings of the models.

4.4. CodecSUPERB Evaluation

While the above evaluation is conducted over VERSA-supported audio quality metrics, we also submit the pre-trained 16kHz codec results to the CodecSUPERB benchmark (hidden sets) [97] for further evaluation on unknown hidden sets using a range of pre-trained models. In the application evaluation, resynthesized audio is input to pre-trained task-specific models for ASR, SPK, emotion recognition (ER), and audio event classification (AEC), with WER, equal error rate (EER), and accuracy (ACC) as evaluation metrics, respectively.

As presented in Table 7, Encodec trained on base LibriTTS has achieved the best performance across most of the speech-related metrics, while SoundStream trained on AMUSE performs better on audio-related metrics. Although large-scale training with speech, audio, and music contributes to the performance of SoundStream and DAC, scaling up training does not necessarily improve the performance of Encodec, especially for speech-related metrics. This finding highlights the need for additional strategies when increasing data size and suggests that existing model architectures may exhibit severe sensitivity to diverse scenarios.

4.5. Downstream Integration: ASR

Setups: The discrete ASR experiments are conducted on LibriSpeech 960 [98], evaluating WER on test-{clean, other} sets. Following [9], the discrete codecs are first converted to embeddings by looking up the codebook from the codec models as input to the system. The downstream ASR models and training hyperparameters are aligned with the configurations in [9].

Results and Discussion: The results are shown in Table 8 (Col. 2).

Table 8: Comparison of codecs on ASR, TTS, SPK, SSE, SVS, and SSL (evaluated on ASR and SLU). + stands for codecs trained on AMUSE. Test-{clean / other} sets are reported for ASR. Models that failed to train are marked with \times .

Model	ASR	NAR-TTS			AR-TTS			SPK	SSE				SVS			SSL (ASR)	SSL (SLU)
	WER ↓	WER ↓	UTMOS ↑	SPK-SIM ↑	WER ↓	UTMOS ↑	SPK-SIM ↑	EER ↓	PESQ ↑	STOI ↑	DNSMOS ↑	MCD ↓	SACC ↑	SingMOS ↑	WER ↓	ACC ↑	
SoundStream Encodec DAC	3.7 / 11.1	3.4	2.34	0.58	6.7	3.70	0.63	27.5	1.79	0.73	0.93	-	-	-	✗	✗	
	3.6 / 10.0	4.3	2.35	0.59	7.7	3.85	0.63	15.7	1.85	0.76	0.95	-	-	-	✗	1.5	
	3.6 / 10.3	5.2	2.12	0.54	10.2	3.75	0.66	29.5	1.73	0.74	0.87	-	-	-	✗	✗	
SoundStream+ Encodec+ DAC+	3.7 / 10.2	4.7	1.84	0.57	9.8	2.97	0.61	15.9	1.76	0.73	0.81	8.82	0.60	2.92	✗	✗	
	3.9 / 10.3	5.4	1.92	0.58	8.6	2.32	0.62	14.1	1.24	0.51	0.62	8.51	0.58	2.86	28.1	68.9	
	4.1 / 10.9	6.2	1.82	0.56	15.9	2.94	0.64	24.6	2.00	0.78	0.87	9.26	0.49	2.58	✗	3.6	

The best performance is achieved by Encodec trained on LibriTTS. For AMUSE-based codecs, SoundStream performs the best. While the performance indicating data scaling for codecs is not always applicable to downstream tasks, it also suggests a weak correlation between audio quality metrics and downstream performance.

4.6. Downstream Integration: TTS

Setups: We use LJSpeech [99] for NAR-TTS and LibriTTS [84] for AR-TTS. Results are reported in WER by Whisper, UTMOS, and SPK-SIM on the test-clean set. We follow [57] for the NAR-TTS, including an encoder and multiple decoders to predict 8 stream codes in parallel. For AR-TTS, we follow VALL-E architecture, including an AR decoder and a NAR decoder. The detailed configurations are generally aligned with [16].

Results and Discussion: The results are shown in Table 8 (Col. 3-8). For NAR-TTS, the model trained with LibriTTS-based SoundStream codec achieves the best intelligibility while the model trained with LibriTTS-based Encodec codec achieves the best UTMOS. For both NAR-TTS and AR-TTS, the performance achieved by LibriTTS-based codecs consistently outperforms their AMUSE counterparts, which should be more attributed to the consistency of dataset between codec and the downstream TTS training.

4.7. Downstream Integration: SSE

Setups: We evaluate the model on WSJ0-2Mix with 8kHz sampling rate data by resampling the audio to the 16kHz used by the codec. Instead of discrete codes, the pre-trained codec encoder and decoder are frozen to serve the SSE system. For the separator and other training configurations, we use identical configurations as in [74]. Models are all trained for 10 epochs using SI-SDR loss. Eight codec streams are used in the experiments.

Results and Discussion: We report the results in Table 8 (Col. 10-12). Different codec types exhibit varying behaviors in SSE performance. While DAC trained on AMUSE has the best PESQ and STOI scores, Encodec trained on LibriTTS achieves the highest DNSMOS. The data scaling from LibriTTS to AMUSE improves DAC performance but does not necessarily enhance the performance of SoundStream and Encodec.

4.8. Downstream Integration: SVS

Setups: We use Openpop [92] for SVS. Following [76, 77], we predict 32 codec streams in parallel, with the same hyperparameters as [76]. Instead of re-training a vocoder, we use the codec decoder for waveform generation. As LibriTTS models do not consider singing, we only evaluate AMUSE models. We evaluate the system based on MCD, semitone accuracy (SACC), and SingMOS [52].

Results and Discussion: We presents the results in Table 8 (Col. 13-15). Although SoundStream has a worse MCD score than Encodec, both SACC and SingMOS show that it is a better choice overall. The result, however, exhibits mismatches between sound quality measures in Table 6, indicating that SVS performance not only depends

on the upper bound set by the codec resynthesis quality but is also related to the difficulties of predicting codec tokens.

4.9. Downstream Integration: SPK

Setups: Our experiments for the SPK task utilize the VoxCeleb1 [100] corpus. The model is trained using the development set, and performance is reported using the EER on the test set.

Results and Discussion: Col. 9 of Table 8 presents the speaker verification performance. Codec tokens trained on the larger AMUSE dataset consistently outperform those trained on LibriTTS. These results confirm that the discretized codec tokens encapsulate speaker information. However, significant overfitting was observed, indicating the need for future research in codec-based speaker verification.

4.10. Downstream Integration: SSL

Setups: We conduct SSL pre-training using LibriSpeech-960 [98]. We perform downstream evaluations on the SSL models using LibriLight Limited [101] for ASR and SLURP [102] for spoken language understanding (SLU). We train variations of HuBERT Base [1] where the model input uses the quantized neural codec tokens from codec models instead of raw waveforms. We initialize embeddings from 8 codebooks and sum those embeddings across the codebooks. After pre-training, we directly fine-tune the models on each downstream task.

Results and Discussion: The results of the fine-tuned SSL models are shown in Table 8. We find instability when using quantized codecs as inputs during the pre-training stage, resulting in failed training even after hyperparameter tuning for certain codecs. Aside from downstream performance, another important aspect is the pre-training speed. Our codec-input SSL models only required 200 GPU hours of pre-training, whereas a model using waveform inputs required 280 hours in an equivalent data setting.

5. CONCLUSION

This paper presents ESPnet-Codec, a framework focusing on neural codec training and evaluation in diverse scenarios. Alongside ESPnet-Codec, we introduce VERSA, a unified audio evaluation toolkit designed for comprehensive speech evaluation. Given the recent advancements in neural codecs, we demonstrate the integration of ESPnet-Codec into six downstream tasks. This integration not only showcases the potential of ESPnet-Codec in various applications but also provides a deeper understanding of neural codecs through extensive probing and analysis.

6. ACKNOWLEDGEMENT

This work used the Bridges2 at PSC and Delta at NCSA through allocations CIS210014 and IRI120008P from the ACCESS program, supported by NSF grants #2138259, #2138286, #2138307, #2137603, and #2138296. We also appreciate Wangyou Zhang’s advice on VERSA implementation.

7. REFERENCES

- [1] W.-N. Hsu *et al.*, “HuBERT: Self-supervised speech representation learning by masked prediction of hidden units,” *TASLP*, 2021.
- [2] S. Chen *et al.*, “WavLM: Large-scale self-supervised pre-training for full stack speech processing,” *JSTSP*, 2022.
- [3] C.-C. Chiu *et al.*, “Self-supervised learning with random-projection quantizer for speech recognition,” in *Proc. ICML*, 2022.
- [4] J. Shi *et al.*, “Multi-resolution HuBERT: Multi-resolution speech self-supervised learning with masked unit prediction,” in *Proc. ICLR*, 2024.
- [5] S.-W. Yang *et al.*, “SUPERB: Speech Processing Universal Performance Benchmark,” in *Proc. Interspeech*, 2021.
- [6] J. Shi *et al.*, “ML-SUPERB: Multilingual Speech Universal Performance Benchmark,” in *Proc. Interspeech*, 2023.
- [7] H.-J. Chang *et al.*, “DistillHuBERT: Speech representation learning by layer-wise distillation of hidden-unit bert,” in *Proc. ICASSP*, 2022.
- [8] X. Chang *et al.*, “Exploration of Efficient End-to-End ASR using Discretized Input from Self-Supervised Learning,” in *Proc. Interspeech*, 2023.
- [9] X. Chang *et al.*, “Exploring speech recognition, translation, and understanding with discrete speech units: A comparative study,” in *Proc. ICASSP*, 2024.
- [10] J. Shi *et al.*, “Bridging speech and textual pre-trained models with unsupervised ASR,” in *Proc. ICASSP*, 2023.
- [11] T. Hayashi *et al.*, “Discretalk: Text-to-speech as a machine translation problem,” *arXiv:2005.05525*, 2020.
- [12] A. Lee *et al.*, “Direct speech-to-speech translation with discrete units,” in *Proc. ACL*, 2022.
- [13] J. Shi *et al.*, “Enhancing speech-to-speech translation with multiple TTS targets,” in *Proc. ICASSP*, 2023.
- [14] B. Yan *et al.*, “ESPnet-ST-v2: Multipurpose spoken language translation toolkit,” in *Proc. ACL*, 2023.
- [15] Y. Yang *et al.*, “Towards universal speech discrete tokens: A case study for ASR and TTS,” in *Proc. ICASSP*, 2024.
- [16] C. Wang *et al.*, “Neural codec language models are zero-shot text to speech synthesizers,” *arXiv:2301.02111*, 2023.
- [17] J. Copet *et al.*, “Simple and controllable music generation,” in *Proc. NeurIPS*, 2023.
- [18] X. Zhang *et al.*, “SpeechTokenizer: Unified speech tokenizer for speech language models,” in *Proc. ICLR*, 2024.
- [19] D. Yang *et al.*, “Uniaudio: An audio foundation model toward universal audio generation,” in *ICML*, 2024.
- [20] D. Zhang *et al.*, “SpeechGPT: Empowering large language models with intrinsic cross-modal conversational abilities,” in *Proc. EMNLP*, 2023.
- [21] X. Chang *et al.*, “The Interspeech 2024 challenge on speech processing using discrete units,” in *Proc. Interspeech*, 2024.
- [22] N. Zeghidour *et al.*, “SoundStream: An end-to-end neural audio codec,” *TASLP*, 2021.
- [23] A. Défossez *et al.*, “High fidelity neural audio compression,” *TMLR*, 2023.
- [24] R. Kumar *et al.*, “High-fidelity audio compression with improved RVQGAN,” *Proc. NeurIPS*, 2024.
- [25] S. Watanabe *et al.*, “ESPnet: End-to-end speech processing toolkit,” in *Proc. Interspeech*, 2018.
- [26] T. Hayashi *et al.*, “ESPnet-TTS: Unified, reproducible, and integratable open source end-to-end text-to-speech toolkit,” in *Proc. ICASSP*, 2020.
- [27] T. Hayashi *et al.*, “ESPnet2-TTS: Extending the edge of TTS research,” *arXiv:2110.07840*, 2021.
- [28] J. Shi *et al.*, “Muskits: An end-to-end music processing toolkit for singing voice synthesis,” in *Proc. Interspeech*, 2022.
- [29] J.-w. Jung *et al.*, “ESPnet-SPK: Full pipeline speaker embedding toolkit with reproducible recipes, self-supervised front-ends, and off-the-shelf models,” in *Proc. Interspeech*, 2024.
- [30] Y. J. Lu *et al.*, “ESPnet-SE++: Speech enhancement for robust speech recognition, translation, and understanding,” in *Proc. Interspeech*, 2022.
- [31] C. Li *et al.*, “ESPnet-SE: End-to-end speech enhancement and separation toolkit designed for asr integration,” in *Proc. SLT*, 2021.
- [32] W. Chen *et al.*, “Reducing barriers to self-supervised learning: HuBERT pre-training with academic compute,” in *Proc. Interspeech*, 2023.
- [33] Y.-C. Wu *et al.*, “Audiodec: An open-source streaming high-fidelity neural audio codec,” in *Proc. ICASSP*, 2023.
- [34] D. Yang *et al.*, “HiFi-codec: Group-residual vector quantization for high fidelity audio codec,” *arXiv:2305.02765*, 2023.
- [35] Z. Du *et al.*, “FunCodec: A fundamental, reproducible and integratable open-source toolkit for neural speech codec,” in *Proc. ICASSP*, 2024.
- [36] X. Zhang *et al.*, “Amphion: An open-source audio, music and speech generation toolkit,” *arXiv:2312.09911*, 2023.
- [37] A. W. Rix *et al.*, “Perceptual evaluation of speech quality (PESQ)—a new method for speech quality assessment of telephone networks and codecs,” in *Proc. ICASSP*, 2001.
- [38] C. H. Taal *et al.*, “An algorithm for intelligibility prediction of time-frequency weighted noisy speech,” *TASLP*, 2011.
- [39] J. Le Roux *et al.*, “SDR—half-baked or well done?” In *Proc. ICASSP*, 2019.
- [40] A. Hines *et al.*, “ViSQOL: An objective speech quality model,” *JASL*, 2015.
- [41] W. Liu *et al.*, “A high fidelity and low complexity neural audio coding,” *arXiv:2310.10992*, 2023.
- [42] Y. Li *et al.*, “Real-time speech frequency bandwidth extension,” in *Proc. ICASSP*, 2021.
- [43] A. Razavi *et al.*, “Generating diverse high-fidelity images with VQ-VAE-2,” *Proc. NeurIPS*, 2019.
- [44] R. Yamamoto *et al.*, “Parallel WaveGAN: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram,” in *Proc. ICASSP*, 2020.
- [45] A. Mustafa *et al.*, “StyleMelGAN: An efficient high-fidelity adversarial vocoder with temporal adaptive normalization,” in *Proc. ICASSP*, 2021.
- [46] J. Kong *et al.*, “HiFi-GAN: Generative adversarial networks for efficient and high fidelity speech synthesis,” *Proc. NeurIPS*, 2020.
- [47] C. Boeddeker *et al.*, “Convolutional transfer function invariant SDR training criteria for multi-channel reverberant speech separation,” in *Proc. ICASSP*, 2021.
- [48] T. Saeki *et al.*, “SpeechBERTScore: Reference-aware automatic evaluation of speech generation leveraging nlp evaluation metrics,” *arXiv:2401.16812*, 2024.
- [49] C. K. Reddy *et al.*, “DNSMOS: A non-intrusive perceptual objective speech quality metric to evaluate noise suppressors,” in *Proc. ICASSP*, 2021.
- [50] T. Saeki *et al.*, “UTMOS: Utokyo-sarulab system for voicemos challenge 2022,” *Interspeech*, 2022.
- [51] L. Diener *et al.*, “PLCMOS—a data-driven non-intrusive metric for the evaluation of packet loss concealment algorithms,” *Interspeech*, 2022.

- [52] Y. Tang *et al.*, “SingMOS: An extensive open-source singing voice dataset for mos prediction,” *arXiv:2406.10911*, 2024.
- [53] A. Radford *et al.*, “Robust speech recognition via large-scale weak supervision,” in *Proc. ICML*, 2023.
- [54] T. Wang *et al.*, “Viola: Unified codec language models for speech recognition, synthesis, and translation,” *arXiv:2305.16107*, 2023.
- [55] Q. Chen *et al.*, “Loss masking is not needed in decoder-only transformer for discrete-token-based ASR,” in *Proc. ICASSP*, 2024.
- [56] L. Ye *et al.*, “ASQ: An ultra-low bit rate ASR-oriented speech quantization method,” *IEEE Signal Processing Letters*, 2023.
- [57] J. Shi *et al.*, “MMM: Multi-layer multi-residual multi-stream discrete speech representation from self-supervised learning model,” in *Proc. Interspeech*, 2024.
- [58] A. Gupta *et al.*, “Exploring the limits of decoder-only models trained on public speech recognition corpora,” *arXiv:2402.00235*, 2024.
- [59] Y. Ren *et al.*, “FastSpeech 2: Fast and high-quality end-to-end text to speech,” in *Proc. ICLR*, 2020.
- [60] J. Kim *et al.*, “Conditional variational autoencoder with adversarial learning for end-to-end text-to-speech,” in *Proc. ICML*, 2021.
- [61] J. Shen *et al.*, “Natural TTS synthesis by conditioning wavenet on mel spectrogram predictions,” in *Proc. ICASSP*, 2018.
- [62] X. Wang *et al.*, “Speechx: Neural codec language model as a versatile speech transformer,” *arXiv:2308.06873*, 2023.
- [63] P. Anastassiou *et al.*, “Seed-TTS: A family of high-quality versatile speech generation models,” *arXiv:2406.02430*, 2024.
- [64] Z. Ju *et al.*, “Naturalspeech 3: Zero-shot speech synthesis with factorized codec and diffusion models,” in *ICML*, 2024.
- [65] D. Yang *et al.*, “Simplespeech: Towards simple and efficient text-to-speech with scalar latent transformer diffusion models,” *arXiv:2406.02328*, 2024.
- [66] J.-w. Jung *et al.*, “Pushing the limits of raw waveform speaker recognition,” in *Proc. Interspeech*, 2022.
- [67] B. Desplanques *et al.*, “ECAPA-TDNN: Emphasized channel attention, propagation and aggregation in TDNN based speaker verification,” in *Proc. Interspeech*, 2020.
- [68] K. C. Puvvada *et al.*, “Discrete audio representation as an alternative to mel-spectrograms for speaker and speech recognition,” in *Proc. ICASSP*, 2024.
- [69] Z. Wang *et al.*, “TF-GridNet: Integrating full- and sub-band modeling for speech separation,” *TASLP*, 2022.
- [70] J. Richter *et al.*, “Speech enhancement and dereverberation with diffusion-based generative models,” *TASLP*, 2023.
- [71] S. Zhao *et al.*, “MossFormer2: Combining transformer and rnn-free recurrent network for enhanced time-domain monaural speech separation,” in *Proc. ICASSP*, 2024.
- [72] J. Q. Yip *et al.*, “SPGM: Prioritizing local features for enhanced speech separation performance,” in *Proc. ICASSP*, 2024.
- [73] H. Song *et al.*, “Exploring WavLM on speech enhancement,” in *Proc. SLT*, 2023.
- [74] J. Q. Yip *et al.*, “Towards audio codec-based speech separation,” in *Proc. Interspeech*, 2024.
- [75] C. Subakan *et al.*, “Exploring self-attention mechanisms for speech separation,” *TASLP*, 2022.
- [76] Y. Wu *et al.*, “Toksing: Singing voice synthesis based on discrete tokens,” in *Proc. Interspeech*, 2024.
- [77] Y. Tang *et al.*, “SingOMD: Singing oriented multi-resolution discrete representation construction from speech models,” in *Proc. Interspeech*, 2024.
- [78] Y. Zhang *et al.*, “VISinger2: High-Fidelity End-to-End Singing Voice Synthesis Enhanced by Digital Signal Processing Synthesizer,” in *Proc. Interspeech*, 2023.
- [79] J. Shi *et al.*, “Sequence-to-sequence singing voice synthesis with perceptual entropy loss,” in *Proc. ICASSP*, 2021.
- [80] T.-Q. Lin *et al.*, “Melhubert: A simplified hubert on mel spectrograms,” in *Proc. ASRU*, 2023.
- [81] T. Parcollet *et al.*, “On the (in) efficiency of acoustic feature extractors for self-supervised speech representation learning,” in *Proc. Interspeech*, 2023.
- [82] L. Lugo *et al.*, “Towards efficient self-supervised representation learning in speech processing,” in *Proc. EACL*, 2024.
- [83] G. Yang *et al.*, “Fast-Hubert: An efficient training framework for self-supervised speech representation learning,” in *Proc. ASRU*, 2023.
- [84] H. Zen *et al.*, “LibriTTS: A corpus derived from librispeech for text-to-speech,” in *Proc. Interspeech*, 2019.
- [85] Y. Shi *et al.*, “AISHELL-3: A multi-speaker mandarin tts corpus,” in *Proc. Interspeech*, 2021.
- [86] J. Shi *et al.*, “Highland Puebla Nahuatl speech translation corpus for endangered language documentation,” in *Proc. AmericasNLP*, 2021.
- [87] J. Shi *et al.*, “Leveraging end-to-end ASR for endangered language documentation: An empirical study on Yolóxochitl Mixtec,” in *Proc. EACL*, 2021.
- [88] R. Huang *et al.*, “Multi-singer: Fast multi-singer singing voice vocoder with a large-scale corpus,” in *Proc. ACMMM*, 2021.
- [89] S. Dai *et al.*, “Singstyle111: A multilingual singing dataset with style transfer,” in *Proc. ISMIR*, 2023.
- [90] L. Zhang *et al.*, “M4singer: A multi-style, multi-singer and musical score provided mandarin singing corpus,” *Proc. NeurIPS*, 2022.
- [91] I. Ogawa *et al.*, “Tohoku Kiritan singing database: A singing database for statistical parametric singing synthesis using japanese pop songs,” *AST*, 2021.
- [92] Y. Wang *et al.*, “Opencpop: A high-quality open source chinese popular song corpus for singing voice synthesis,” in *Proc. Interspeech*, 2022.
- [93] J. Shi *et al.*, “Singing voice data scaling-up: An introduction to ACE-Opencpop and ACE-KiSing,” in *Proc. Interspeech*, 2024.
- [94] J. Koguchi *et al.*, “PJS: Phoneme-balanced japanese singing-voice corpus,” in *Proc. APSIPA ASC*, 2020.
- [95] S. Takamichi *et al.*, “JSUT and JVS: Free japanese voice corpora for accelerating speech synthesis research,” *AST*, 2020.
- [96] J. Liu *et al.*, “Diffsinger: Singing voice synthesis via shallow diffusion mechanism,” in *Proc. AAAI*, 2022.
- [97] H. Wu *et al.*, “Codec-SUPERB: An in-depth analysis of sound codec models,” *arXiv:2402.13071*, 2024.
- [98] V. Panayotov *et al.*, “Librispeech: An ASR corpus based on public domain audio books,” in *Proc. ICASSP*, 2015.
- [99] K. Ito *et al.*, *The lj speech dataset*, <https://keithito.com/LJ-Speech-Dataset/>, 2017.
- [100] A. Nagrani *et al.*, “VoxCeleb: A large-scale speaker identification dataset,” in *Proc. Interspeech*, 2017.
- [101] J. Kahn *et al.*, “Libri-Light: A benchmark for ASR with limited or no supervision,” in *Proc. ICASSP*, 2020.
- [102] E. Bastianelli *et al.*, “SLURP: A spoken language understanding resource package,” in *Proc. EMNLP*, 2020.