ESC: Efficient Speech Coding with Cross-Scale Residual Vector Quantized Transformers

Yuzhe Gu[1,2], **Enmao Diao**[2]
[1]University of Pennsylvania, Philadelphia, PA
[2]Duke University, Durham, NC
tracygu@seas.upenn.edu enmao.diao@duke.edu

Abstract

Neural speech codecs aim to compress input signals into minimal bits while maintaining content quality in a low-latency manner. However, existing neural codecs often trade model complexity for reconstruction performance. These codecs primarily use convolutional blocks for feature transformation, which are not inherently suited for capturing the local redundancies in speech signals. To compensate, they require either adversarial discriminators or a large number of model parameters to enhance audio quality. In response to these challenges, we introduce the **E**fficient **S**peech **C**odec (ESC)[1], a lightweight, parameter-efficient speech codec based on a cross-scale residual vector quantization scheme and transformers. Our model employs mirrored hierarchical window transformer blocks and performs step-wise decoding from coarse-to-fine feature representations. To enhance bitrate efficiency, we propose a novel combination of vector quantization techniques along with a pre-training paradigm. Extensive experiments demonstrate that ESC can achieve high-fidelity speech reconstruction with significantly lower model complexity, making it a promising alternative to existing convolutional audio codecs.

1 Introduction

Recent advancements in deep learning have demonstrated the superiority of neural speech codecs over traditional ones, which rely on complex expert design and psycho-acoustic knowledge (Valin et al., 2012; Dietz et al., 2015). Early efforts integrating deep generative models, such as WaveNet (Oord et al., 2016) and SampleRNN (Mehri et al., 2017), into audio codecs have delivered promising results. These models, acting as powerful decoders, synthesize high-quality speech from intermediate representations produced by traditional codecs (Kleijn et al., 2018; Klejsa et al., 2019). However, their auto-regressive nature of the decoding process often introduces significant inference latency, limiting their practical application.

Alternatively, some end-to-end neural audio codecs leverage the vector quantization (VQ) network first introduced by Van Den Oord et al. (2017). VQ networks use a learnable collection of codevectors, known as a codebook, to quantize continuous vectors by assigning them to the nearest codeword. This discretization positions VQNs well-suited for both generation and compression tasks. Following this approach, existing VQ codecs (Zeghidour et al., 2021; Défossez et al., 2023; Kumar et al., 2023) typically employ a three-stage architecture: a convolutional encoder and decoder, and a residual vector quantization (RVQ) module (Vasuki and Vanathi, 2006) applied in the latent space. The encoder and decoder downsample and upsample audio waveform features, creating hierarchical representations. RVQ further refines vanilla vector quantization by minimizing quantization error through a series of codebooks that recursively quantize the residuals from previous stages. Additionally, these codecs employ adversarial discriminators to remove artifacts and produce high-fidelity audio reconstructions. Substantial effort has been dedicated to designing effective audio discriminators, including an improved feature matching loss (Kumar et al., 2019), as well as various multi-resolution waveform and spectrogram discriminators (Kong et al., 2020; Zeghidour et al., 2021; Défossez et al., 2023; gil Lee et al., 2023). VQ-based audio codecs have demonstrated remarkable performance in audio reconstruction, even at ultra-low bitrates.

Despite these advantages, we find that convolutional VQ codecs heavily depend on powerful discriminators to produce high-quality audio, posing additional optimization challenges due to adversarial training. Moreover, these codecs tend

[1]Code and pretrained models available at https://github.com/yzGuu830/efficient-speech-codec

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 10084–10096

to confront computational constraints, as they require a large number of parameters to balance high compression rates and reconstruction performance. To address these issues, our work develops a more parameter-efficient speech codec by reducing model complexity and implementing the following architectural improvements: 1) replacing convolutional layers with efficient Swin-Transformer Blocks (STBs) (Liu et al., 2021), which can better model acoustic features; 2) utilizing the cross-scale residual vector quantization (CS-RVQ) scheme (Jiang et al., 2022a) instead of RVQ, extending quantization from a fixed level to multiple levels.

In addition, training VQ codecs frequently leads to a significant challenge: *codebook collapse*, where a fraction of the codebook remains underutilized in representing input vectors. This issue is frequently observed when training visual tokenizers for generative vision tasks (Takida et al., 2022; Zhang et al., 2023; Huh et al., 2023). To address this problem in speech compression, we propose combining product vector quantization (PVQ) (Baevski et al., 2019), code factorization (Yu et al., 2022), and Euclidean normalization (Łańcucki et al., 2020) to enhance codebook utilization. Furthermore, we introduce a learning paradigm to facilitate optimization, which includes a pre-training stage where the codebooks are deactivated and trained subsequently.

In summary, the key contributions of our work are as follows:

- We introduce ESC, a fully transformer-based speech codec with cross-scale quantization structures. It achieves a superior tradeoff between compression rate, reconstruction quality, and model complexity, outperforming current state-of-the-art models.
- We propose a novel combination of vector quantization techniques within the cross-scale residual vector quantization (CS-RVQ) framework, coupled with a pre-training paradigm that effectively mitigates codebook collapse and enhances bitrate efficiency.
- Extensive comparisons with Descript's audio codec on a multilingual speech corpus demonstrate that transformers and CS-RVQ, the core components of ESC, are superior backbones for speech foundation models than the mainstream convolutions and RVQ.

2 Related Work

2.1 Neural Audio Codecs

Recently, most notable neural audio codecs have been based on the vector quantization (VQ) network, including SoundStream (Zeghidour et al., 2021), EnCodec (Défossez et al., 2023), and Descript's audio codec (DAC) (Kumar et al., 2023). SoundStream is distinguished as the first universal codec capable of handling diverse audio types. EnCodec improves compression rates by integrating a lightweight transformer language model within the discrete latent space and implements a streaming architecture. Building on similar backbones, Kumar et al. (2023) further explore the implications of quantization dropout, a technique for bitrate scalability, and demonstrate the superiority of periodic inductive bias functions over common activation functions for audio signal modeling (gil Lee et al., 2023; Ziyin et al., 2020). These models directly process audio waveforms and are classified as time-domain codecs.

In contrast, frequency-domain codecs focus on processing more intuitive audio spectrogram features. Lyra (Kleijn et al., 2021), for example, converts audio waveforms into log mel-spectrograms and directly quantizes them into tokens. Due to the non-invertibility of mel-spectrograms, it relies on a vocoder (Kalchbrenner et al., 2018) for waveform synthesis. To circumvent the inefficiencies associated with heavy vocoders, some frequency-domain codecs, including TFNet (Jiang et al., 2022b) and our ESC, employ the invertible Short-time Fourier Transform (STFT) to convert waveforms into complex spectra. This design enables the reconstructed STFT spectra to be seamlessly inverted back into waveforms without information loss using inverse-STFT. Among recent audio codecs, DAC achieves state-of-the-art compression ratios and reconstruction quality, though its computation bottlenecks are sometimes overlooked.

2.2 Swin Transformers

Vision Transformers (ViTs) (Dosovitskiy et al., 2020) have outperformed convolutional neural networks (CNNs) in various image processing tasks, largely due to their superior ability to capture complex patterns. The Swin Transformer (Liu et al., 2021), a notable variant, enhances this capability by employing a hierarchical approach with shifted window attention mechanisms, enabling it to scale efficiently to high-resolution signals while main-

taining computational efficiency. In the context of image compression, Swin Transformers have demonstrated exceptional performance. Studies by Zhu et al. (2021) and Zou et al. (2022) show that Swin Transformers surpass CNNs in modeling spatial hierarchies and long-range dependencies. The attention mechanism facilitates the accurate preservation of essential details and textures, even at lower bitrates. These capabilities suggest that transformers could also be effective in applications beyond image compression, such as modeling audio spectrograms.

2.3 Vector Quantization

In the Vector Quantized Variational Autoencoder (VQ-VAE) (Van Den Oord et al., 2017), vector quantization (VQ) functions as a trainable layer that deterministically quantizes encoded latent variables by mapping them to their nearest neighbors in an embedding codebook. A VQ layer, denoted as $Q(\cdot\,;\mathcal{C})$, is parameterized by a collection of continuous vectors $\mathcal{C} = \{\boldsymbol{c}_1, ..., \boldsymbol{c}_K\}$, each referred to as a codeword, with its associated index known as a code. The layer quantizes a vector $\boldsymbol{z}_e \in \mathbb{R}^d$ to $\boldsymbol{z}_q \in \mathbb{R}^d$ by selecting the Euclidean nearest codeword $\boldsymbol{c}_k$ from the codebook $\mathcal{C}$, i.e.,

$$\boldsymbol{z}_q := \boldsymbol{c}_k = \underset{\boldsymbol{c}_j \in \mathcal{C}}{\arg\min} ||\boldsymbol{z}_e - \boldsymbol{c}_j||_2^2. \quad (1)$$

For convenience, we denote the output of the VQ function as $(\boldsymbol{z}_q, \tilde{z}_q)$, where $\tilde{z}_q := k$ represents the discrete code corresponding to the nearest codeword. During compression, the encoding process outputs the discrete index $\tilde{z}_q$, which is stored with a $\log_2 K$ bit budget. The decoding process starts by retrieving the continuous vector $\boldsymbol{z}_q$ from the codebook using the index $\tilde{z}_q$. The VQ function $Q(\cdot;\mathcal{C})$ is non-differentiable due to the $\arg\min$ operator. Common strategies use a straight-through estimator (STE) (Bengio et al., 2013) to bypass this in back-propagation. In other words, the gradient component $\frac{\partial \boldsymbol{z}_q}{\partial \boldsymbol{z}_e}$ is estimated by identity. Additionally, auxiliary losses including codebook loss and commitment loss are proposed to pull the codewords and latent features closer:

$$\mathcal{L}_{vq} = ||\mathrm{sg}(\boldsymbol{z}_e) - \boldsymbol{z}_q||_2^2 + \beta||\boldsymbol{z}_e - \mathrm{sg}(\boldsymbol{z}_q)||_2^2. \quad (2)$$

Here $\mathrm{sg}(\cdot)$ denotes the stop-gradient operator. The first term updates the codebook with an l_2 error, pushing the codewords towards the input vectors. The second term ensures that $\boldsymbol{z}_e$ commits to the embedding without growing arbitrarily. The scalar β balances the importance of updating the codebook and the encoder.

2.4 Codebook Collapse

Straight-through estimators (STEs) can lead to significant issues, most notably *codebook collapse*, as detailed by Vuong et al. (2023). In a recent study, Huh et al. (2023) provide a plausible explanation, attributing the collapse to an internal codebook covariate shift during training. Frequent adjustments in encoder representations cause misalignment with the codebook, resulting in only a subset of codewords being updated. Consequently, VQ layers are prone to divergence, often ending up with a significant number of inactive vectors. Various strategies have been proposed in generative modeling context to address this issue, including stochastic quantization (Takida et al., 2022; Zhang et al., 2023), self-annealed soft-to-hard quantization (Agustsson et al., 2017), re-initializing codewords using K-means centroids every few epochs (Łańcucki et al., 2020; Dhariwal et al., 2020), and reformulating with finite scalar quantization (Mentzer et al., 2024). In audio compression, Kumar et al. (2023) address codebook collapse by down-projecting codewords (Yu et al., 2022) and normalizing them within a Euclidean ball (Łańcucki et al., 2020).

3 Efficient Speech Codec (ESC)

3.1 Overall Architecture

As illustrated in Figure 1, ESC operates on the complex spectrum $\mathcal{X} \in \mathbb{R}^{2\times F\times T}$ derived from the Short-Time Fourier Transform (STFT) of a speech signal. Here, the real and imaginary components of $\mathcal{X}$ are treated as separate channels. Instead of using strided convolutions, ESC comprises a series of mirrored transformer encoder and decoder layers, each performing downsampling or upsampling to create coarse and fine representations, as described in Section 3.3. Starting from the quantized latents at the bottleneck VQ, the decoder progressively reconstructs the original spectrum by leveraging multi-level quantized residuals between the intermediate features of the encoder and decoder. This cross-scale decoding mechanism is further detailed in Section 3.4. Finally, the reconstructed spectrum $\hat{\mathcal{X}}$ is transformed back into a waveform through the inverse-STFT.

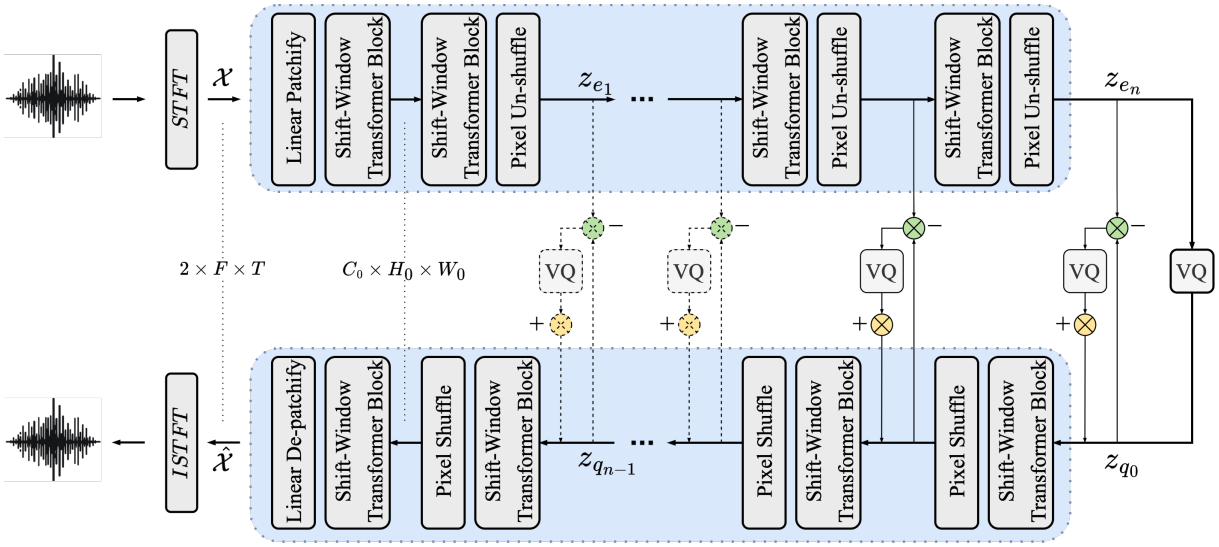

Figure 1: The framework of ESC: input speech is transformed to a complex STFT $\mathcal{X}$ and linearly embedded into patches. Encoder STBs iteratively halve the frequency resolution and produce hierarchical feature representations. Mirrored decoder STBs recover the frequency resolution by progressively leveraging coarse-to-fine quantized residual features between encoder and decoder hidden states. The entire network is solely composed of efficient transformer blocks and vector quantization layers. The figure displays a scenario when the deepest 3 of $n+1$ total bitstreams (solid lines) are transmitted, with others left inactive.

3.2 Notations

We first define some notations for clarity. The encoder and decoder are denoted by $F_\phi(\cdot)$ and $G_\psi(\cdot)$, respectively, each being a composition of individual layer functions $f_{\phi_1}, \ldots, f_{\phi_n}$ and $g_{\psi_1}, \ldots, g_{\psi_n}$. We use $\mathcal{Z} \in \mathbb{R}^{C \times F \times T}$ to denote a spectrum feature and $\boldsymbol{z} \in \mathbb{R}^{CF}$ to denote a flattened time frame vector in $\mathcal{Z}$. Specifically, $\mathcal{Z}_{e_i}$ refers to the feature after the i-th encoder layer, and $\mathcal{Z}_{q_i}$ denotes the i-th decoder feature.

$$\mathcal{Z}_{e_i} = f_{\phi_i} \circ \ldots \circ f_{\phi_1} \circ \mathcal{Z}_{e_0} \tag{3}$$

$$\mathcal{Z}_{q_i} = g_{\psi_i} \circ \ldots \circ g_{\psi_1} \circ \mathcal{Z}_{q_0}, \tag{4}$$

Here, $\mathcal{Z}_{e_0}$ is the original input feature and $\mathcal{Z}_{q_0}$ is the latent representation at the bottleneck.

3.3 Transformer Encoder and Decoder

To effectively capture redundancies within audio signals, we replace convolutional layers with hierarchical Swin Transformer blocks (STBs) and their extended decoding counterparts.

Patchify. The encoder starts with a linear patchify module, where the complex spectrum $\mathcal{X}$ is divided into small patches and linearly up-projected:

$$\mathcal{X} \in \mathbb{R}^{2 \times F \times T} \xrightarrow{\text{Patchify}} \mathcal{Z}_{e_0} \in \mathbb{R}^{C_0 \times H_0 \times W_0}. \tag{5}$$

Here, the patch size across the frequency and temporal dimensions is $(\frac{F}{H_0}, \frac{T}{W_0})$. This step reduces the input resolution to alleviate the computational burden on attention computation. At the end of the decoder, a symmetric de-patchify module reshapes the decoded patch feature $\mathcal{Z}_{q_n}$ and linearly down-projects it to produce a recovered spectrum $\hat{\mathcal{X}}$.

Swin Transformer blocks. STBs in both the encoder and decoder employ window-based multi-head self-attention (W-MSA), partitioning spectrum features into smaller windows and computing attention in parallel within each window. This approach enables more efficient computation compared to vanilla attention mechanisms. To ensure connections between windows, STBs cascade two interleaved W-MSAs, with the outputs of the first being shifted for the second. This design allows STBs to capture local and global feature dependencies both effectively and efficiently.

Downsampling and upsampling. ESC maintains temporal resolution while scaling frequency resolution to equalize bitrates across different bitstreams. To achieve this, we modify the original patch merging/splitting modules with a single-dimensional pixel unshuffle/shuffle module (Shi et al., 2016) along the frequency dimension. During encoder downsampling, an intermediate encoder spectrum feature $\mathcal{Z}_{e_i} \in \mathbb{R}^{C_i \times H_i \times W_i}$ is first reshaped and then projected by $P_{e_i} \in \mathbb{R}^{vC_i \times C_{i+1}}$ as follows:

$$\xrightarrow{\text{reshape}} \mathbb{R}^{vC_i \times \frac{H_i}{v} \times W_i} \xrightarrow{\text{proj}} \mathbb{R}^{C_{i+1} \times \frac{H_i}{v} \times W_i}, \tag{6}$$

where v is the down-scaling factor. The upsampling process mirrors this operation in reverse. An intermediate decoder feature $\mathcal{Z}_{q_i} \in \mathbb{R}^{C_i \times H_i \times W_i}$ is first projected by $P_{q_i} \in \mathbb{R}^{C_i \times vC_{i+1}}$ and then reshaped, resulting in an up-scaled frequency resolution:

$$\xrightarrow{\text{proj}} \mathbb{R}^{vC_{i+1} \times H_i \times W_i} \xrightarrow{\text{reshape}} \mathbb{R}^{C_{i+1} \times vH_i \times W_i}. \quad (7)$$

Overall, the transformer encoder and decoder layers are mirrored, creating symmetric and hierarchical representations of the input audio spectrum. With these backbones, ESC is a fully transformer-based codec without any convolutional modules.

3.4 Cross-Scale Residual Vector Quantization

To achieve parameter-efficient modeling of audio signals, ESC employs multi-scale features that capture coarse-to-fine information. It integrates the more intuitive residual-based cross-scale vector quantization (CS-RVQ) framework proposed by Jiang et al. (2022a), eliminating the need for additional networks to merge encoder and decoder features for improved reconstruction quality. As depicted in Algorithm 1, Algorithm 2 and Figure 1, the decoding process is conditioned on the encoded quantized residuals between encoder and decoder features from low-to-high resolution scales. This approach differs from the commonly used residual vector quantization scheme, which operates solely at the lowest scale, relying on the highest-level information while overlooking low-level details.

Encoding. The encoding process begins with the encoder $F_\phi(\cdot)$, creating multi-scale encoder features $\boldsymbol{z}_{e_1}, ..., \boldsymbol{z}_{e_n}$. $\boldsymbol{z}_{e_n}$ is first quantized by the bottleneck quantizer Q_0 to form the lowest bitstream. This represents the simplest case when the number of transmitted bitstream s is set to 1, and CS-RVQ reduces to a fixed-scale VQ at the bottleneck. For higher bitstreams, the residual between symmetric encoder and decoder at higher resolutions, $\boldsymbol{z}_{e_{n-i+1}} - \boldsymbol{z}_{q_{i-1}}$, is quantized by Q_i. The quantized residual $\boldsymbol{q}_i$ is then added back to $\boldsymbol{z}_{q_{i-1}}$ and decoded by the subsequent decoder layer $g_{\phi_i}(\cdot)$, producing the next decoder feature $\boldsymbol{z}_{q_i}$. Recursively, residuals at higher resolutions are progressively quantized, forming the remaining bitstreams (see Algorithm 1, Lines 3-6). This mechanism enables multi-scale learning, allowing the decoder layers to incrementally reduce quantization errors by conditioning on encoder-decoder residual features. When $s > 2$, this encoding process requires forward passing $s-2$ additional decoder layers to produce residuals at higher levels. After encoding, the input $\boldsymbol{z}_{e_0}$ is compressed into multi-level codes $\tilde{z}_{q_0}, \tilde{z}_{q_1}, \ldots, \tilde{z}_{q_{s-1}}$.

Algorithm 1 CS-RVQ Encoding

Require: A flattened time frame $\boldsymbol{z}_{e_0} \in \mathbb{R}^{C_0 H_0}$, encoder $F_\phi(\cdot)$, decoder $G_\psi(\cdot)$, vector quantizers $Q_0, Q_1, ..., Q_{s-1}$, number of bitstreams s

1: $\boldsymbol{z}_{e_1}, ..., \boldsymbol{z}_{e_n} \leftarrow F_\phi(\boldsymbol{z}_{e_0})$ ▷ Encoder forward pass
2: $\boldsymbol{z}_{q_0}, \tilde{z}_{q_0} \leftarrow Q_0(\boldsymbol{z}_{e_n})$ ▷ bottom VQ
3: **for** $i = 1 \ldots s-2$ **do**
4: $\quad \boldsymbol{q}_i, \tilde{z}_{q_i} \leftarrow Q_i(\boldsymbol{z}_{e_{n-i+1}} - \boldsymbol{z}_{q_{i-1}})$
5: $\quad \boldsymbol{z}_{q_i} \leftarrow g_{\psi_i}(\boldsymbol{z}_{q_{i-1}} + \boldsymbol{q}_i)$
6: **end for** ▷ Encoding involves $s-2$ decoder layers
7: **if** $s > 1$ **then**
8: $\quad \boldsymbol{q}_{s-1}, \tilde{z}_{q_{s-1}} \leftarrow Q_i(\boldsymbol{z}_{e_{n-s+2}} - \boldsymbol{z}_{q_{s-2}})$
9: **end if**
10: **return** $\tilde{z}_{q_0}, \tilde{z}_{q_1}, ..., \tilde{z}_{q_{s-1}}$

Algorithm 2 CS-RVQ Decoding

Require: Codes $\tilde{z}_{q_0}, \tilde{z}_{q_1}, ..., \tilde{z}_{q_{s-1}}$, decoder $G_\psi(\cdot)$, vector quantizers $Q_0, Q_1, ..., Q_{s-1}$

1: $\boldsymbol{z}_{q_0} \xleftarrow{Q_0} \tilde{z}_{q_0}$ ▷ Retrieve codewords from bottom VQ
2: **for** $i = 1 \ldots s-1$ **do**
3: $\quad \boldsymbol{q}_i \xleftarrow{Q_i} \tilde{z}_{q_i}$
4: $\quad \boldsymbol{z}_{q_i} \leftarrow g_{\psi_i}(\boldsymbol{z}_{q_{i-1}} + \boldsymbol{q}_i)$
5: **end for** ▷ Decoding refined by quantized residuals
6: **for** $i = s \ldots n$ **do**
7: $\quad \boldsymbol{z}_{q_i} \leftarrow g_{\psi_i}(\boldsymbol{z}_{q_{i-1}})$
8: **end for** ▷ Continue with regular decoding
9: **return** $\boldsymbol{z}_{q_n}$

Decoding. The decoding process starts by retrieving the quantized latent at the bottom VQ using code $\tilde{z}_{q_0}$, which provides the initial decoder input $\boldsymbol{z}_{q_0}$. At higher levels, the codes $\tilde{z}_{q_1}, ..., \tilde{z}_{q_{s-1}}$ are iteratively used to retrieve codewords, producing multi-scale low-to-high quantized residuals $\boldsymbol{q}_1, ..., \boldsymbol{q}_{s-1}$. In Algorithm 2, Lines 2-5, each quantized residual $\boldsymbol{q}_i$ is added back to the corresponding decoder feature $\boldsymbol{z}_{q_{i-1}}$ to refine the decoding process. Starting from the s-th decoder layer, there are no quantized residuals, and the remaining layers perform regular decoding. Finally, the recovered frame vector $\boldsymbol{z}_{q_n}$ is obtained, benefiting from $s-1$ quantized residual features.

Training. During training, the encoding and decoding processes are concatenated to form a complete forward pass. To enable bitrate scalability, we sample $s \sim \text{Uniform}\{1, \ldots, n\}$ at a rate p within

each training mini-batch. p is a hyperparameter that balances the reconstruction quality at different bitrates, as proposed by Kumar et al. (2023).

3.5 Mitigating Codebook Collapse

ESC performs a per-frame vector quantization. Before nearest neighbor searching, each input spectrum frame feature in $\mathcal{Z}$ needs to be flattened, merging the frequency and channel dimensions. This approach can result in large input vector dimensions for VQ, increasing the optimization challenges associated with codebook underutilization.

Vector quantization setups. To optimize the codebooks effectively, we modify the vanilla VQ by combining product vector quantization with code-vector factorization at each bitstream. Specifically, a flattened d-dimensional frame vector $\boldsymbol{z}_{e_i}$ is split into a set of l sub-vectors. Each sub-vector $\boldsymbol{z}_{e_i}^{(m)}$ is down-projected by $W_{\text{in}} \in \mathbb{R}^{\frac{d}{l} \times u}$, where $u \ll d$, and then quantized using an individual codebook $\mathcal{C}_m$. The selected code-vector is then up-projected by $W_{\text{out}} \in \mathbb{R}^{u \times \frac{d}{l}}$ to form $\boldsymbol{z}_{q_i}^{(m)}$:

$$\boldsymbol{z}_{e_i} \equiv \{\boldsymbol{z}_{e_i}^{(m)} \mid \boldsymbol{z}_{e_i}^{(m)} \in \mathbb{R}^{\frac{C_i H_i}{l}}, m = 1, ..., l\}, \quad (8)$$

$$\boldsymbol{z}_{q_i}^{(m)} = W_{\text{out}}^{\top} \arg\min_{\boldsymbol{c}_j \in \mathcal{C}_m} ||W_{\text{in}}^{\top} \boldsymbol{z}_{e_i}^{(m)} - \boldsymbol{c}_j||_2. \quad (9)$$

Additionally, both the projected vector $W_{\text{in}}^{\top} \boldsymbol{z}_{e_i}^{(m)}$ and codebook $\mathcal{C}_m$ are l_2 normalized before computing the distance matrix. This equalizes the scales of input vectors and codewords, enhancing codebook optimization by allowing a larger subset of codewords to receive gradients (Łańcucki et al., 2020).

Pre-training paradigm. Training transformers can be challenging, and jointly training them with VQ layers is even more difficult. To address this, we propose a pre-training paradigm that includes a warm-start to facilitate the learning process. Initially, all VQ layers are deactivated, meaning no quantization occurs. During this "pre-training" stage, only the encoder and decoder are updated within the CS-RVQ framework, allowing latent features to bypass the quantizers and flow directly into the decoder layers. Once the encoder and decoder have converged by minimizing reconstruction objectives, we resume training the entire VQ codec as usual. This approach helps mitigate the distribution shift of encoder representations by pre-optimizing the encoder. It helps stabilize codebook training and improve bitrate efficiency. Moreover, pre-training an auto-encoder is simpler, as it avoids the quantization errors associated with VQs. The detailed algorithm is provided in Appendix A.

3.6 Training Objectives

To train our codec, we use a combination of reconstruction loss $\mathcal{L}_{recon}$ and vector quantization loss $\mathcal{L}_{vq}$. The reconstruction loss, $\mathcal{L}_{recon}$, consists of two components: an l_2 distance between the complex spectrum $\mathcal{X}$ and its reconstruction $\hat{\mathcal{X}}$, which forces the model to reconstruct the real and imaginary parts, weighted by λ_1, and a multi-scale mel-spectrogram loss (Kumar et al., 2023), weighted by λ_2. These are denoted as $\mathcal{L}_{stft}$ and $\mathcal{L}_{mel}$:

$$\mathcal{L}_{recon} = \lambda_1 \mathcal{L}_{mel} + \lambda_2 \mathcal{L}_{stft}. \quad (10)$$

$\mathcal{L}_{vq}$ comprises the standard codebook and commitment losses as described in Equation 2. It is averaged across the l product vector quantizers and summed over all s bitstreams. The final objective for joint optimization is the summation of $\mathcal{L}_{recon}$ and $\mathcal{L}_{vq}$. To deactivate the VQ layers during the pre-training stage, $\mathcal{L}_{vq}$ is set to zero.

4 Experiments

4.1 Experimental Setup

Datasets. We extract 150 hours of 16kHz multilingual clean speech from the DNS Challenge dataset (Reddy et al., 2021). Training samples are clipped into 3-second segments, and validation samples into 10-second segments. For evaluation, we compile 1158 multilingual 10-second speech clips with non-overlapping speakers from the LibriSpeech (Panayotov et al., 2015), Multilingual LibriSpeech (Pratap et al., 2020), and AIShell (Shi et al., 2020) datasets.

Baselines. We compare our ESC against the current state-of-the-art time-domain codec DAC, by reproducing three versions[2] on our dataset:

1) DAC-Base (adversarial): Descript's original released codec, operating on 16kHz audio signals. It has 74M parameter count in total. Its associated discriminator has 42M additional parameter count.

2) DAC-Tiny (adversarial): A smaller version of DAC-Base, with reduced encoder and decoder dimensions, for a fair comparison with ESC.

3) DAC-Tiny (non-adversarial): A smaller and non-adversarial version of DAC to assess the impact of discriminators on improving audio fidelity.

[2]Reproduction settings are detailed in Appendix B.1.

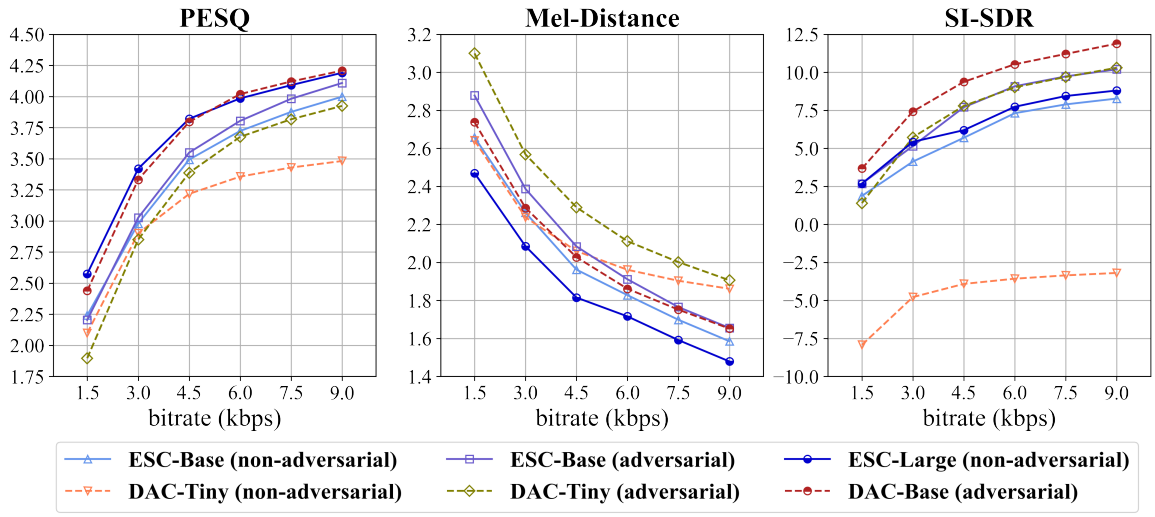

Figure 2: Reconstruction quality evaluation of different baseline codecs: dashed lines represent DAC baselines and solid lines represent our ESC models, with x-axis being transmission bits per second and y-axis being PESQ (↑), Mel-Distance (↓) and SI-SDR(↑). The metrics are averaged over our composed 1158 10-second speech clips.

Implementation details. Similar to DAC baselines, we provide different versions of ESC[3]:
1) ESC-Base (non-adversarial): A base version codec consisting of 6 encoder/decoder layers, with bitrates ranging from 1.5 to 9.0 kbps. It contains 8.39M parameters when operating at 9.0 kbps.
2) ESC-Base (adversarial): An adversarial version using the same multi-scale multi-band waveform and spectrogram discriminator in DAC.
3) ESC-Large (non-adversarial): A scaled-up version with increased Swin Transformer layer depth, having 15.58M parameters at 9.0 kbps.
Our ESC variants are trained using the AdamW optimizer (Loshchilov, 2017) with a learning rate of 1e-4 and a weight decay of 1e-2. Training runs up to 0.4 million iterations without learning rate schedulers. The proposed pre-training phase consists of 0.75 million iterations. After pre-training, the codebooks are initialized with a Kaiming normalization distribution (He et al., 2015). The quantization dropout rate p is set to 0.75. Loss weighting hyperparameters are set as $\lambda_1 = 0.25$, $\lambda_2 = 1.0$, and the commitment loss weighting $\beta = 0.25$. For ESC-Base (adversarial), the $\mathcal{L}_{stft}$ component is eliminated. We use the HingeGAN (Lim and Ye, 2017) adversarial loss formulation and the l_1 feature matching loss (Kumar et al., 2019), following the approach of DAC.
Automatic evaluation metrics. We use objective metrics to efficiently evaluate reconstruction performance. These include the PESQ score (Union, 2007) from the speech enhancement domain, following Jiang et al. (2022a); the l_1 distance between log mel-spectrograms of reference and decoded waveforms (Mel-Distance) (Kumar et al., 2023); and the scale-invariant source-to-distortion ratio (SI-SDR) (Le Roux et al., 2019). To measure codec inference latency, we use the real-time factor (RTF), defined as the ratio of speech audio duration to model processing time (Défossez et al., 2023).

[3]Complete configurations are detailed in Appendix B.2.

Codec	Bitrate	#Param.	Real Time Factor ↑ Enc.	Dec.
ESC-Base	3.0 kbps	8.10M	33.66	34.97
	6.0 kbps	8.21M	27.84	33.02
	9.0 kbps	8.39M	24.45	33.95
DAC-Tiny	3.0 kbps	7.96M	42.26	49.52
	6.0 kbps	8.07M	44.66	48.63
	9.0 kbps	8.17M	43.00	49.10
ESC-Large	3.0 kbps	15.30M	17.91	20.81
	6.0 kbps	15.41M	15.48	19.87
	9.0 kbps	15.58M	13.73	20.56
DAC-Base	3.0 kbps	73.99M	12.77	3.36
	6.0 kbps	74.15M	11.43	3.13
	9.0 kbps	74.31M	11.81	3.25

Table 1: Complexity evaluation results of different baseline codecs: RTFs are measured from 100 10-second speech clips on an Intel Xeon Platinum 8352V CPU.

4.2 Comparison with DAC

We provide a thorough comparison focusing on compression rate, reconstruction quality, and inference efficiency, as shown in Table 1 and Figure 2.
Performance evaluation. First, it is important

to note that ESC-Base and DAC-Tiny are similar in model size, each with approximately 8 million trainable parameters. Our results show that ESC-Base consistently outperforms DAC-Tiny across all bitrates, even without an adversarial discriminator. In contrast, DAC-Tiny's reconstruction quality significantly drops without a discriminator in training, particularly in SI-SDR statistics. This indicates a heavy reliance of DAC models on GANs for maintaining high reconstruction quality. Notably, ESC-Base is compatible with the same convolution-based GAN discriminator used in DAC, as evidenced by its improved performance across all metric curves in its adversarial variant. Additionally, ESC-Large demonstrates that increasing ESC's model size can further enhance performance, with its PESQ curve matching that of DAC-Base, the top-performing and largest model. While DAC-Base achieves higher SI-SDR values, ESC-Large records a smaller Mel-Distance. Thus, we conclude that the two codecs achieve comparable performance, even though ESC-Large is trained without an adversarial discriminator.

Complexity evaluation. Despite their exceptional performance, Descript's top-performing codecs face significant computational challenges. This is evident from Table 1, where the decoding real-time factor (RTF) for DAC-Base is approximately 3.0, making it rather impractical for real-time applications. In contrast, our transformer-based ESC achieves much higher decoding RTFs (approximately 34), indicating superior computational efficiency. Although ESC-Base is not as fast as DAC-Tiny due to the overhead of attention computation, it offers substantially better speech reconstruction capabilities, striking a favorable balance between compression performance and computation latency. Future work could incorporate transformer speedup techniques such as FlashAttention (Dao et al., 2022) to further enhance ESC's latency further. Moreover, following the CS-RVQ scheme, ESC possesses faster encoding speeds at lower bitrates—a capability not evidently found in DAC models.

These results suggest that our transformer-based codec, equipped with CS-RVQ, is a more parameter-efficient foundation model compared to time-domain convolutional counterparts. ESC is shown to be a more lightweight and effective neural speech codec, as ESC-Large achieves comparable performance to DAC-Base without the need for a powerful discriminator. Specifically, it boasts approximately $\times$4.8 smaller model size, $\times$1.4 faster encoding speed, and $\times$6.4 faster decoding speed.

Method	Bitrate	PESQ ↑	Mel dist. ↓	SI-SDR ↑	VQ util. ↑
CNN + RVQ	3.0 kbps	2.71	2.82	0.57	96.8%
	6.0 kbps	2.93	2.69	1.03	98.2%
	9.0 kbps	2.96	2.68	1.05	98.7%
CNN + CS-RVQ	3.0 kbps	2.70	2.81	2.19	96.6%
	6.0 kbps	3.47	2.41	3.79	97.7%
	9.0 kbps	3.75	2.25	4.16	97.3%
SwinT + RVQ	3.0 kbps	2.97	2.22	0.77	**98.1%**
	6.0 kbps	3.14	2.08	1.35	**99.0%**
	9.0 kbps	3.16	2.07	1.39	**99.2%**
ESC-Base (SwinT + CS-RVQ)	3.0 kbps	3.07	**2.21**	**3.55**	97.8%
	6.0 kbps	**3.73**	**1.80**	**4.74**	98.3%
	9.0 kbps	**3.92**	**1.62**	**5.33**	97.9%
ESC-Base w/o Pre-training	3.0 kbps	**3.09**	2.25	1.75	97.7%
	6.0 kbps	3.53	1.97	2.87	98.1%
	9.0 kbps	3.58	1.89	2.88	86.5%

Table 2: Performance evaluation of different ablation models: results are obtained from the 1157 10-second speech clips in our test dataset.

4.3 Ablation Study

To investigate the effectiveness of the proposed components in ESC, we conducted thorough ablation experiments[4] by training frequency-domain codecs operating on complex STFT spectra with different architectures. For fair comparisons, all other ablation models listed in Table 2 have similar model sizes to ESC-Base.

Swin Transformers and CNNs. To demonstrate that transformers are superior auto-encoder backbones in neural speech coding, we focus on two pairs of experiments: CNN/SwinT + RVQ and CNN/SwinT + CS-RVQ. In these experiments, the channel dimensions of the CNN blocks are set to match the hidden dimensions of the Swin Transformer Blocks (STBs). The comparison, as shown in Table 2, reveals that transformer-based codecs consistently outperform CNN-based codecs across all performance metrics and bitrates, regardless of the quantization scheme used.

CS-RVQ and RVQ. Table 2 highlights that CS-RVQ is a superior quantization scheme compared to RVQ, regardless of whether the backbone is CNN or STB. RVQ-based codecs hit performance bottlenecks, as adding more VQs does not improve audio quality (*e.g.*, from 6.0 kbps to 9.0 kbps). However, codecs using the CS-RVQ scheme do not face such bottlenecks at higher bitrates and consistently outperform their RVQ counterparts. CS-RVQ is therefore a superior vector quantization framework that leverages multi-scale features effectively.

[4]Implementation setups are detailed in Appendix B.3.

Effect of pre-training paradigm. To evaluate the efficacy of the pre-training stage, we conducted an experiment of ESC-Base w/o pre-training. We monitored the VQ utilization rate, calculated as the sum of entropy (in bits) divided by the maximum number of bits from all transmitted bitstreams. This metric reflects bitrate efficiency and the fraction of seldom-used codewords. The results indicate that models with pre-training achieve a near 1.0 utilization rate. However, ESC-Base w/o pre-training displays a lower utilization rate at 9.0 kbps, and its reconstruction performance is also inferior to that of the fully pre-trained ESC-Base. These findings suggest that the pre-training paradigm indeed helps avoid bitrate wastage and improve audio reconstruction quality.

5 Conclusions

In this paper, we introduce ESC, the first fully transformer-based neural speech foundation model designed for multilingual speech coding. ESC surpasses existing state-of-the-art time-domain VQ-based codecs in terms of complexity and achieves comparable compression performance without the need for a powerful adversarial discriminator. Our extensive evaluations demonstrate that the cross-scale residual vector quantization scheme and the Swin Transformer backbones are better suited for neural speech coding than the convolutional blocks and residual vector quantization utilized in mainstream codecs. Overall, our study suggests a promising direction for speech foundation models. Future research could focus on expanding multi-scale vector quantization techniques and investigating additional transformer variants optimized for speech signal modeling.

6 Limitations

First, recent neural audio codecs are increasingly utilized in downstream generation tasks, where the codec acts as a foundation model to create discrete acoustic representations (Borsos et al., 2023; Kreuk et al., 2022; Siuzdak, 2023; Wang et al., 2023; Du et al., 2024). These compressed representations, treated as acoustic tokens, are suitable for auto-regressive language modeling in generative tasks. However, our work does not explore this important aspect. A promising future direction would be to evaluate ESC in downstream applications such as speech synthesis and speech recognition. We anticipate that the cross-scale code representations learned from transformer backbones could offer advantages over the fixed-scale features of mainstream convolutional codecs in these tasks.

Second, different automatic metrics for audio evaluation can produce inconsistent results, which is evidenced in our results. To further strengthen our conclusions, it is necessary to conduct subjective evaluations involving human evaluators, such as MUSHRA listening tests (Series, 2014). Despite this limitation, we provide a collection of demo speech samples publicly available in our codebase, which we hope will help demonstrate ESC's performance and compensate for the absence of subjective metrics.

Besides, the primary focus of this work is to demonstrate the superiority of transformer and cross-scale frameworks over other mainstream methods, rather than to develop a production-ready codec like DAC or EnCodec. Nonetheless, given the scalability of transformers (Kaplan et al., 2020), increasing the ESC model size and training it on larger and more diverse audio datasets also represent a promising direction for enhancing its practical applicability.

Finally, as discussed in Section 3.4, the cross-scale residual vector quantization (CS-RVQ) scheme requires the partial use of decoder layers during the encoding process, introducing additional latency as the bitrate increases. Similar to residual vector quantization, CS-RVQ requires careful sampling of the transmitted bitstream during training to achieve scalable bitrates within a single model. This sampling strategy can lead to performance trade-offs across different bitrates and may cause instability during training. Therefore, future research in speech foundational models could explore leveraging alternative recurrent structures (Toderici et al., 2017; Johnston et al., 2018; Diao et al., 2020) to improve coding scalability and address these challenges.

References

Eirikur Agustsson, Fabian Mentzer, Michael Tschannen, Lukas Cavigelli, Radu Timofte, Luca Benini, and Luc V Gool. 2017. Soft-to-hard vector quantization for end-to-end learning compressible representations. *Advances in neural information processing systems*, 30.

Alexei Baevski, Steffen Schneider, and Michael Auli. 2019. vq-wav2vec: Self-supervised learning of discrete speech representations. *arXiv preprint arXiv:1910.05453*.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

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*.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359.

Alexandre Défossez, Jade Copet, Gabriel Synnaeve, and Yossi Adi. 2023. High fidelity neural audio compression. *Transactions on Machine Learning Research*. Featured Certification, Reproducibility Certification.

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*.

Enmao Diao, Jie Ding, and Vahid Tarokh. 2020. Drasic: Distributed recurrent autoencoder for scalable image compression. In *2020 Data Compression Conference (DCC)*, pages 3–12. IEEE.

Martin Dietz, Markus Multrus, Vaclav Eksler, Vladimir Malenovsky, Erik Norvell, Harald Pobloth, Lei Miao, Zhe Wang, Lasse Laaksonen, Adriana Vasilache, et al. 2015. Overview of the evs codec architecture. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5698–5702. IEEE.

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

Zhihao Du, Shiliang Zhang, Kai Hu, and Siqi Zheng. 2024. Funcodec: A fundamental, reproducible and integrable open-source toolkit for neural speech codec. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 591–595. IEEE.

Sang gil Lee, Wei Ping, Boris Ginsburg, Bryan Catanzaro, and Sungroh Yoon. 2023. BigVGAN: A universal neural vocoder with large-scale training. In *The Eleventh International Conference on Learning Representations*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

Minyoung Huh, Brian Cheung, Pulkit Agrawal, and Phillip Isola. 2023. Straightening out the straight-through estimator: Overcoming optimization challenges in vector quantized networks. In *International Conference on Machine Learning*, pages 14096–14113. PMLR.

Xue Jiang, Xiulian Peng, Huaying Xue, Yuan Zhang, and Yan Lu. 2022a. Cross-scale vector quantization for scalable neural speech coding. In *Interspeech 2022*, pages 4222–4226.

Xue Jiang, Xiulian Peng, Chengyu Zheng, Huaying Xue, Yuan Zhang, and Yan Lu. 2022b. End-to-end neural speech coding for real-time communications. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 866–870.

Nick Johnston, Damien Vincent, David Minnen, Michele Covell, Saurabh Singh, Troy Chinen, Sung Jin Hwang, Joel Shor, and George Toderici. 2018. Improved lossy image compression with priming and spatially adaptive bit rates for recurrent networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4385–4393.

Nal Kalchbrenner, Erich Elsen, Karen Simonyan, Seb Noury, Norman Casagrande, Edward Lockhart, Florian Stimberg, Aaron Oord, Sander Dieleman, and Koray Kavukcuoglu. 2018. Efficient neural audio synthesis. In *International Conference on Machine Learning*, pages 2410–2419. PMLR.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

W Bastiaan Kleijn, Felicia SC Lim, Alejandro Luebs, Jan Skoglund, Florian Stimberg, Quan Wang, and Thomas C Walters. 2018. Wavenet based low rate speech coding. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 676–680. IEEE.

W Bastiaan Kleijn, Andrew Storus, Michael Chinen, Tom Denton, Felicia SC Lim, Alejandro Luebs, Jan Skoglund, and Hengchin Yeh. 2021. Generative speech coding with predictive variance regularization. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6478–6482. IEEE.

Janusz Klejsa, Per Hedelin, Cong Zhou, Roy Fejgin, and Lars Villemoes. 2019. High-quality speech coding with sample rnn. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7155–7159. IEEE.

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

Felix Kreuk, Gabriel Synnaeve, Adam Polyak, Uriel Singer, Alexandre Défossez, Jade Copet, Devi Parikh, Yaniv Taigman, and Yossi Adi. 2022. Audiogen: Textually guided audio generation. *arXiv preprint arXiv:2209.15352*.

Kundan Kumar, Rithesh Kumar, Thibault De Boissiere, Lucas Gestin, Wei Zhen Teoh, Jose Sotelo, Alexandre De Brebisson, Yoshua Bengio, and Aaron C Courville. 2019. Melgan: Generative adversarial networks for conditional waveform synthesis. *Advances in neural information processing systems*, 32.

Rithesh Kumar, Prem Seetharaman, Alejandro Luebs, Ishaan Kumar, and Kundan Kumar. 2023. High-fidelity audio compression with improved RVQGAN. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Adrian Łańcucki, Jan Chorowski, Guillaume Sanchez, Ricard Marxer, Nanxin Chen, Hans JGA Dolfing, Sameer Khurana, Tanel Alumäe, and Antoine Laurent. 2020. Robust training of vector quantized bottleneck models. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE.

Jonathan Le Roux, Scott Wisdom, Hakan Erdogan, and John R Hershey. 2019. Sdr–half-baked or well done? In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 626–630. IEEE.

Jae Hyun Lim and Jong Chul Ye. 2017. Geometric gan. *arXiv preprint arXiv:1705.02894*.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

I Loshchilov. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Soroush Mehri, Kundan Kumar, Ishaan Gulrajani, Rithesh Kumar, Shubham Jain, Jose Sotelo, Aaron Courville, and Yoshua Bengio. 2017. SampleRNN: An unconditional end-to-end neural audio generation model. In *International Conference on Learning Representations*.

Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. 2024. Finite scalar quantization: VQ-VAE made simple. In *The Twelfth International Conference on Learning Representations*.

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

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5206–5210. IEEE.

Vineel Pratap, Qiantong Xu, Anuroop Sriram, Gabriel Synnaeve, and Ronan Collobert. 2020. Mls: A large-scale multilingual dataset for speech research. *arXiv preprint arXiv:2012.03411*.

Chandan KA Reddy, Harishchandra Dubey, Vishak Gopal, Ross Cutler, Sebastian Braun, Hannes Gamper, Robert Aichner, and Sriram Srinivasan. 2021. Icassp 2021 deep noise suppression challenge. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6623–6627. IEEE.

B Series. 2014. Method for the subjective assessment of intermediate quality level of audio systems. *International Telecommunication Union Radiocommunication Assembly*.

Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. 2016. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1874–1883.

Yao Shi, Hui Bu, Xin Xu, Shaoji Zhang, and Ming Li. 2020. Aishell-3: A multi-speaker mandarin tts corpus and the baselines. *arXiv preprint arXiv:2010.11567*.

Hubert Siuzdak. 2023. Vocos: Closing the gap between time-domain and fourier-based neural vocoders for high-quality audio synthesis. *arXiv preprint arXiv:2306.00814*.

Yuhta Takida, Takashi Shibuya, WeiHsiang Liao, Chieh-Hsin Lai, Junki Ohmura, Toshimitsu Uesaka, Naoki Murata, Shusuke Takahashi, Toshiyuki Kumakura, and Yuki Mitsufuji. 2022. SQ-VAE: Variational bayes on discrete representation with self-annealed stochastic quantization. In *International Conference on Machine Learning*.

George Toderici, Damien Vincent, Nick Johnston, Sung Jin Hwang, David Minnen, Joel Shor, and Michele Covell. 2017. Full resolution image compression with recurrent neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 5306–5314.

IT Union. 2007. Wideband extension to recommendation p. 862 for the assessment of wideband telephone networks and speech codecs. *International Telecommunication Union, Recommendation P*, 862.

Jean-Marc Valin, Koen Vos, and Timothy Terriberry. 2012. Definition of the opus audio codec. Technical report.

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

A Vasuki and PT Vanathi. 2006. A review of vector quantization techniques. *IEEE Potentials*, 25(4):39–47.

Tung-Long Vuong, Trung Le, He Zhao, Chuanxia Zheng, Mehrtash Harandi, Jianfei Cai, and Dinh Phung. 2023. Vector quantized wasserstein auto-encoder. *arXiv preprint arXiv:2302.05917*.

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*.

Jiahui Yu, Xin Li, Jing Yu Koh, Han Zhang, Ruoming Pang, James Qin, Alexander Ku, Yuanzhong Xu, Jason Baldridge, and Yonghui Wu. 2022. Vector-quantized image modeling with improved VQGAN. In *International Conference on Learning Representations*.

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

Jiahui Zhang, Fangneng Zhan, Christian Theobalt, and Shijian Lu. 2023. Regularized vector quantization for tokenized image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18467–18476.

Yinhao Zhu, Yang Yang, and Taco Cohen. 2021. Transformer-based transform coding. In *International Conference on Learning Representations*.

Liu Ziyin, Tilman Hartwig, and Masahito Ueda. 2020. Neural networks fail to learn periodic functions and how to fix it. *Advances in Neural Information Processing Systems*, 33:1583–1594.

Renjie Zou, Chunfeng Song, and Zhaoxiang Zhang. 2022. The devil is in the details: Window-based attention for image compression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 17492–17501.

A Pre-training Paradigm

The proposed pre-training paradigm for optimizing vector quantization layers is detailed in Algorithm 3. During the pre-training phase, all vector quantization layers are bypassed, effectively reducing the codec to a standard autoencoder trained solely on reconstruction losses (Lines 2-4). Once the encoder and decoder reach a certain level of convergence, the VQ layers are reactivated, and joint optimization resumes. In the pre-training phase, we set the $\arg\min$ nearest-neighbor selection as an identity function, making $\boldsymbol{z}_q$ equal to the input vector $\boldsymbol{z}_e$.

Algorithm 3 Pre-training Paradigm

1: **repeat**
2: $\quad \hat{\mathcal{X}} = G_\psi(F_\phi(\mathcal{X}))$
3: $\quad \mathcal{L} = \mathcal{L}_{recon}(\mathcal{X}, \hat{\mathcal{X}})$
4: $\quad$ take gradient descent step on $\nabla_\phi \mathcal{L}, \nabla_\psi \mathcal{L}$
5: **until** converged
6: activate VQs and continue learning as usual

B Experiment Details

B.1 DAC Reproduction Setups

Our customized reproduction of DAC models closely follows the official development scripts. The original DAC model, designed for 16kHz audio signals, employs 12 VQ layers in its residual VQ module, supporting bitrates ranging from 0.5 kbps to 6.0 kbps. To ensure a fair comparison with ESC at similar bitrate levels, we extended the number of VQ layers in the RVQ module to 18, resulting in DAC-Base. For DAC-Tiny, we reduced the encoder dimension from 64 to 32 and the decoder dimension from 1536 to 288, while keeping other parameters unchanged. All DAC baselines were trained for 0.4 million iterations with a batch size of 16 on our multilingual speech dataset. Additional configuration details can be found in the official release[5].

B.2 ESC Architecture Configurations

Overall, all three ESC variants are trained in distributed setups for 0.4 million iterations across 4 NVIDIA RTX 4090 GPUs with a total batch size

[5]The official configuration for 16kHz DAC model is available at https://github.com/descriptinc/descript-audio-codec/blob/main/conf/final/16khz.yml

of 36. These experiments took approximately 100 GPU hours.

B.2.1 Model Parameters

The parameter configurations for ESC-Base are provided in Table 3. For STFT transformation, we use a 20 ms window length and a 5 ms hop length, implemented with torchaudio. The number of FFT points is set to 382, resulting in a frequency dimension of 192. In the Swin Transformer, the layer depth represents the number of Swin Transformer Blocks (STBs) cascaded at each encoder and decoder layer. We use GELU activation functions and LayerNorm for normalization. In the down-sampling/up-sampling module, we use a scaling factor of $v = 2$ to un-shuffle/shuffle along the frequency resolution only. Before the vector quantization layers, ESC processes two overlapping time frames together. To implement this, the flattened spectrum feature $\mathcal{Z}$ is reshaped from $\mathbb{R}^{W_i \times H_i C_i}$ to $\mathbb{R}^{W_i/2 \times 2H_i C_i}$. Each frame is then split into sub-vectors, down-projected, and l_2 normalized before computing the distance matrix. The VQ layer at each bitstream of ESC-Base consumes $\log_2 1024 \times 3 \times 150 = 4500$ bits per 3-second input speech (*i.e.*, 1.5 kbps bitrate). For the scaled-up ESC-Large variant, we increase the STB layer depth from 2 to 4 while keeping the other configurations unchanged.

Modules	Parameters	Values
STFT	Window/Hop Length	[20ms, 5ms]
	Number of FFT	382
Encoder/Decoder	Patch Size	[3, 2]
	Layer Dims $C_1, ..., C_6$	[45, 72, 96, 144, 192, 384]
	Attention Heads	[3, 3, 6, 12, 24, 24]
	Layer Depth	2
	Scaling Factor v	2
Vector Quantization	Product VQ Size l	3
	Codevector Dimension u	8
	Codebook Size K	1024

Table 3: Parameter configurations of model variant ESC-Base, which comprises 6 encoder/decoder layers.

B.2.2 Adversarial Training Setup

In the ESC-Base (adversarial) variant, the GAN discriminator is identical to the one used in DAC, consisting of a multi-period discriminator (MPD), multi-band discriminator (MBD), and multi-scale STFT discriminator (MSD), totaling over 42 million parameters. The adversarial loss formulation follows the official DAC-Base (adversarial) configuration. Additionally, we maintain the pre-training paradigm for 0.75 million iterations in this variant, with the discriminator intervening in training only after the pre-training stage finishes.

B.3 Details on Ablation Experiments

All ablation models operate on the complex STFT spectrum, as in ESC (SwinT + CS-RVQ), using the same STFT configurations specified in Table 3. These models were trained for 0.25 million iterations, with 0.025 million iterations allocated for pre-training. The Swin Transformer configurations mirror those used in ESC-Base. Similarly, the vector quantization setup in the CS-RVQ models follows that of ESC-Base. In total, the ablation experiments required approximately 80 hours on 4 RTX 4090 GPUs.

B.3.1 Convolution Blocks

For models with CNN backbones, the convolutional channel dimensions were set to match the hidden sizes of the STB-based models. Each CNN block consists of one residual unit and one down-sampling/upsampling 2D convolutional layer with a stride of 2 along the frequency resolution only. The residual unit consists of two 2D convolutional layers, each followed by BatchNorm and Parametric ReLU activation.

B.3.2 Residual Vector Quantization Setups

For models using RVQs, we adapted the basic RVQ framework commonly used in time-domain codecs. To process frequency-domain spectrum features at the latent bottleneck, we combined RVQ with product vector quantization. Specifically, the flattened time frame vector is split into sub-group vectors, which are then recursively quantized, as in standard RVQs. We set the number of product VQs to 3 and the number of residual VQs to 6, ensuring the bitrate levels match those of ESC-Base (1.5 kbps per bitstream, 6 in total).