GS-Codec: A Gaussian-Splatting Bottleneck for Neural Audio Coding

Ron Aluf Alon Canfi Eliya Nachmani
School of Electrical and Computer Engineering
Ben-Gurion University of the Negev
alufr@post.bgu.ac.il, canfia@post.bgu.ac.il, eliyanac@bgu.ac.il

Abstract

Neural audio codecs compress waveforms into compact discrete tokens that underpin speech language models, real-time communication, and large-scale audio storage. Almost every dominant design, including residual vector quantization, finite scalar quantization, and single-codebook variants, follows the VQ-VAE template by partitioning the encoder latent through a learned codebook or a fixed scalar grid. We ask whether this partition is necessary. We introduce GS-Codec, a neural speech codec whose bottleneck is a parametric signal decomposition rather than a quantizer. We adapt Gaussian splatting from 3D scene reconstruction to one-dimensional latents. An inner optimization loop fits each encoder segment as a weighted sum of 1D Gaussian primitives. The decoder then reconstructs the waveform from the rendered sum. To avoid the cost of this iterative inner loop at inference time, we additionally train a lightweight GS Predictor Net that regresses the primitive parameters in a single forward pass. The encoder and decoder are trained end-to-end through the inner loop, with no quantizer anywhere in the training pipeline: the bottleneck is the decomposition itself, and scalar quantization is applied only post-training to the fitted parameters. Rather than relying on discrete codebook stages for bitrate control, our representation exposes a fine-grained rate-quality tradeoff: a single trained checkpoint supports post-training bitrate control by varying the number of primitives and the per-parameter bit depth, with no retraining required. GS-Codec matches or exceeds well-established open-source codecs such as EnCodec and DAC on speaker similarity (SIM), intelligibility (STOI), and perceptual quality (UTMOS) at comparable bitrates, while achieving comparable semantic performance (WER). Code and audio samples are available at https://ronaluf.github.io/gs-codec/.

1 Introduction

In the wake of Large Language Models [8, 84, 18, 78, 79, 61], there is growing demand for unified multimodal architectures and efficient audio compression. Neural audio codecs have emerged as foundational technology, compressing raw waveforms via neural networks [97, 16, 42] and tokenizing them into compact discrete representations that bridge high-rate acoustic signals and transformer-based models [6, 101]. These tokens are essential for TTS [85, 12, 47, 7, 35, 70, 32], music generation [1, 15, 6, 40, 46], voice conversion, dialogue [17, 101, 31, 20], and real-time communication [97, 96, 81]. Token fidelity directly bounds synthesis quality in LM-based TTS [85, 6] and music generation [1, 15], making codecs the critical infrastructure for multimodal audio systems [17, 100, 68].

The dominant architecture [16, 42, 93, 73] pairs a convolutional encoder with a Residual Vector Quantization (RVQ) bottleneck [97] and mirror decoder. RVQ iteratively quantizes residual error across codebook stages, enabling flexible bitrate-quality trade-offs; EnCodec [16] and DAC [42]

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

achieve near-transparent reconstruction at 3-6 kbps. However, these models share a structural assumption: compression must proceed through a learned discrete codebook, committing the latent to a lookup over a finite set of vectors fixed at training time. Bitrate adjusts only in discrete steps defined by the number of codebook stages, and the representation is susceptible to codebook collapse [19, 44, 106], wasting representational capacity. Whether the bottleneck can instead be a continuous parametric decomposition that encodes structure rather than assigns indices remains an open question.

We introduce **GS-Codec**, a neural audio codec that replaces the RVQ quantizer with a Gaussian Splatting (GS) bottleneck. Building on Gaussian splatting's success in image representation and compression [102, 52, 103, 99], GS-Codec fits N_G localized Gaussian primitives to each audio segment's latent via a lightweight per-clip optimizer. Each primitive shares a temporal width and per-channel amplitudes across a local neighbourhood of frames. This overlapping representation provides a structural prior at the bottleneck that framewise quantization does not explicitly impose. Rather than mapping to discrete codes (RVQ) or a fixed scalar grid (FSQ), GS-Codec uses a parametric signal decomposition: the encoder latent is approximated as a weighted sum of Gaussian primitives, with only those parameters scalar-quantized post-training. Since quantization is applied after training, a single checkpoint supports fine-grained post-training bitrate control by varying the number of primitives and per-parameter bit depth, with no retraining required. Experiments on LibriTTS, LJSpeech, and LibriSpeech show GS-Codec matches or exceeds established open-source codecs on perceptual quality and speaker similarity, while opening new design axes for neural audio compression.

Our contributions are as follows:

1. We propose GS-Codec, a neural speech codec whose bottleneck is a 1D Gaussian-splatting decomposition rather than a vector or scalar quantizer. To our knowledge, this is the first neural codec to replace partition-based discretization with a parametric signal decomposition inside the latent bottleneck.
2. We introduce a lightweight GS Predictor Net that learns to approximate the iterative Gaussian Splatting process, amortizing the multi-step optimization loop into a single forward pass for fast inference.
3. We demonstrate fine-grained post-training bitrate control via varying primitive count and bit depth, and report comparisons against open-source speech codecs on LibriTTS, LJSpeech, and LibriSpeech, matching or exceeding baselines on perceptual quality and speaker similarity.

2 Related Work

Neural speech and audio coding follows an encoder-quantiser-decoder template. WaveNet [82] motivated faster alternatives displacing Opus [81] and LPCNet [80]; SoundStream [97], Lyra V2 [96], EnCodec [16], and DAC [42] established RVQ as the dominant bottleneck. Adversarial and diffusion vocoders-HiFi-GAN [38], MelGAN [41], Parallel WaveGAN [92], UnivNet [30], BigVGAN [49], Vocos [72], DiffWave [39]-showed that multi-scale spectrogram losses drive fidelity gains later adopted by codec pipelines [31, 93]. Variants target quantiser structure (HiFi-Codec [93], SuperCodec [105], SNAC [73]), benchmarking [89], semantic alignment [101], single-codebook compression [31], real-time dialogue [17], and dynamic frame rates [51]. Ultra-low-rate work includes DualCodec [50] (12.5 Hz dual-stream), SemantiCodec [53] (diffusion decoder, sub-kbps), TAAE [63] (FSQ, 400-700 bps), TaDiCodec [87] (6.25 Hz, text-guided diffusion), TS3-Codec [90] (pure transformer), X-Codec [95], and NaturalSpeech 3 [32] (factorised VQ).

The VQ bottleneck traces to Gray [26] and VQ-VAE [83, 66], with extensions for collapse [19], robustness [44], residual composition [48, 54], inter-codebook optimisation [104], regularisation [106], variable bitrates [9], and FSQ [57], which quantizes each latent coordinate on a fixed scalar grid instead of a learned codebook-all via straight-through gradients [5]; our bottleneck requires no codebook and no straight-through estimator for discretization, as scalar quantization is applied only post-training to the continuous fitted parameters [55].

Codec tokens underpin language models [6, 85, 12, 1, 15, 40, 7, 47, 20, 68] built on transformers [84], self-supervised encoders [60, 3, 28, 14, 11, 65, 4], and multimodal architectures [100]; token qual-

ity directly bounds downstream fidelity [43]. Cross-codec benchmarks [89, 94, 18] and metrics PESQ [67], UTMOS [69], STOI [75] anchor evaluation.

Our bottleneck is a parametric signal decomposition rather than a quantiser, following the differentiable DSP tradition [23, 27] but applied inside the bottleneck. Gaussian splatting [34, 58] has been extended to 2-D geometry [29], image compression [102, 52, 103, 99, 13], and 1-D time-series [86]; broader radial-basis kernels [64] suggest natural extensions. Feed-forward amortisation [74, 10, 88], implicit compression [22, 21, 77], and Instant-NGP [59] inform our inner-loop design. SIRENs [71] ground per-signal fitting; Lanzendörfer and Wattenhofer [45] apply this to audio by fitting a SIREN per clip-we share that spirit but replace the network with an analytic Gaussian decomposition, making parameters directly quantisable. Rate control contemporaries FlexiCodec [51] and Wav-Tokenizer [31] rely on fixed codebooks; GS-Codec instead varies the resolution of a parametric splat fit. Our SEANet [76] backbone uses the Snake activation [107] and AdamW [56], with a VAE/GAN [37, 25] training skeleton.

3 Method

We propose GS-Codec, a neural audio compression method whose bottleneck is a parametric signal decomposition based on one-dimensional Gaussian splatting. An encoder network, denoted as E and based on the SEANet architecture [97], maps the input waveform to a continuous latent representation. An inner optimization loop then fits this latent as a weighted sum of one-dimensional Gaussian primitives. A SEANet-based decoder network, denoted as D, then reconstructs the audio from this parametric approximation. The entire model is trained end-to-end, with E learning to yield splattable representations and D learning to invert the rendered sum. The overall architecture is depicted in Figure 1.

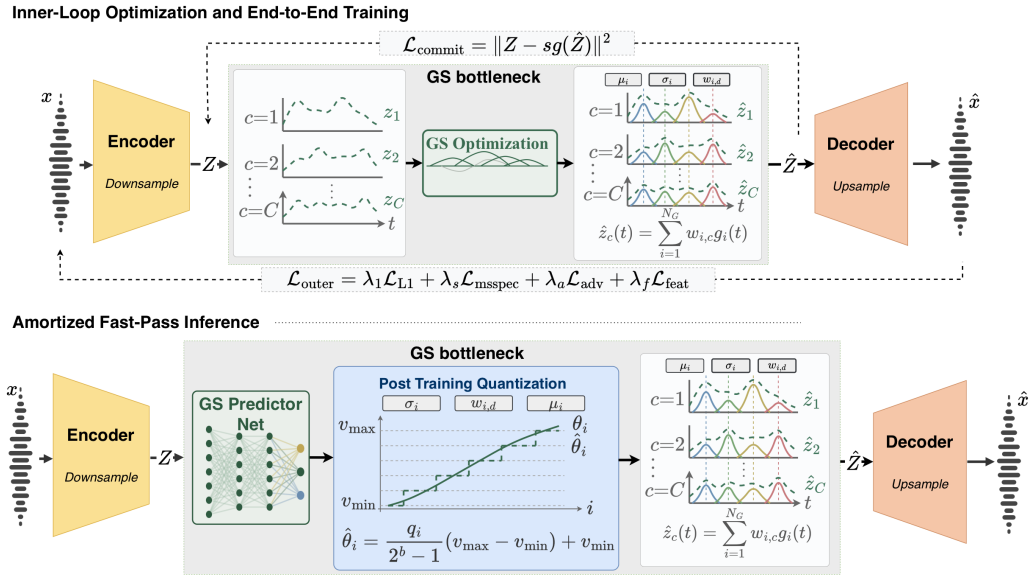


Figure 1: GS-Codec pipeline. **Training** (top): The encoder latent Z is approximated by N_G Gaussian primitives via an inner optimization loop. **Inference** (bottom): A GS Predictor Net regresses primitive parameters in a single pass, which are then quantized, transmitted, and decoded.

3.1 End-to-End Training with the GS Bottleneck

The training pipeline consists of the encoder E, the Gaussian-splatting bottleneck, and the decoder D, optimized jointly via a combination of reconstruction and adversarial losses.

3.1.1 Architecture

Encoder The encoder network E maps a time-domain input waveform $x \in \mathbb{R}^{1\times L}$, where L is the number of audio samples, to a continuous latent tensor $Z \in \mathbb{R}^{C\times T}$, where C is the channel dimension and T is the number of temporal latent frames. The network E follows the SEANet convolutional architecture [97] with the Snake activation [107], progressively downsampling the input waveform through strided convolutions to a lower-rate latent representation. Concrete values for C and other architectural hyperparameters are given in Appendix A.

GS Bottleneck and 1D Gaussian Splatting The core of GS-Codec is the Gaussian-splatting bottleneck. Inspired by 3D Gaussian splatting for scene reconstruction [34], we adapt the formulation to one-dimensional latent fitting. To the best of our knowledge, this is the first application of Gaussian splatting as a compression bottleneck in a neural audio codec. A Gaussian primitive is a spatially localized, one-dimensional basis function parameterized by a temporal center $\mu \in \mathbb{R}$ and a temporal scale $\sigma \in \mathbb{R}^+$. The primitive is defined as $\phi(t;\mu,\sigma) = \exp\left(-\frac{(t-\mu)^2}{2\sigma^2}\right)$. The basis function ϕ is differentiable everywhere and bounded, making it well-suited for gradient-based optimization.

To represent a segment of the latent signal Z, we approximate it as a weighted superposition of N_G Gaussian primitives, where N_G is the number of primitives. Each primitive $i \in \{1,\dots,N_G\}$ is described by its center μ_i, its scale σ_i (which is shared across all C channels), and a per-channel amplitude vector $w_i \in \mathbb{R}^C$. Thus, the rendered approximation for a specific channel $c \in \{1,\dots,C\}$ at time frame $t \in \{0,\dots,T-1\}$ is computed as $\hat{Z}_{c,t} = \sum_{i=1}^{N_G} w_{i,c}\phi(t;\mu_i,\sigma_i)$, which we denote collectively as the rendered tensor $\hat{Z} \in \mathbb{R}^{C\times T}$.

The parameters of these primitives, denoted as $\Theta = \{\mu_i,\sigma_i,w_i\}_{i=1}^{N_G}$, are obtained via a 1D Gaussian splatting inner loop applied during the forward pass. Given a fixed latent Z from the encoder E, we minimize the mean squared error (MSE) loss, defined as

$$\mathcal{L}_{\text{inner}} = \frac{1}{CT}\|Z - \hat{Z}\|_F^2. \tag{1}$$

Starting from an initial state $\Theta^{(0)}$, the parameters are iteratively updated for K gradient-based optimization steps. While any first-order optimizer (e.g., Adam [36]) can be used in practice, a standard gradient descent step takes the form

$$\Theta^{(k)} = \Theta^{(k-1)} - \alpha^{(k)}\nabla_\Theta \mathcal{L}_{\text{inner}}(\Theta^{(k-1)}), \quad k \in \{1,\dots,K\}, \tag{2}$$

where $\alpha^{(k)}$ is the step size. Figure 2 illustrates this convergence on a single latent channel.

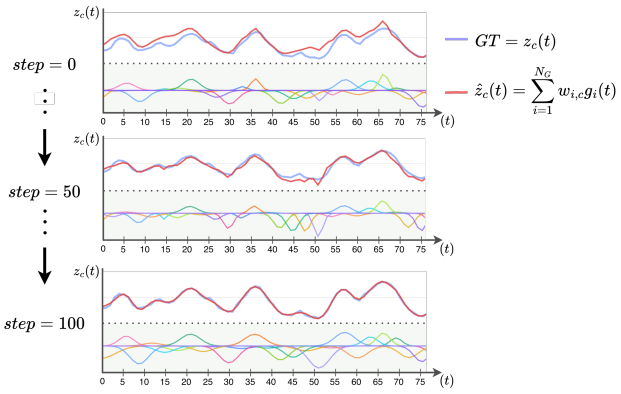


Figure 2: Visualization of the GS inner loop on a single latent channel across optimization steps 0, 50, and 100. Ground-truth Z_c is shown in blue, rendered approximation $\hat{Z}_c$ in red, and individual weighted Gaussian primitives are shown underneath. The horizontal axis denotes the temporal latent frame (t), while the vertical axis represents the latent value.

Decoder The decoder network D mirrors the encoder architecture, applying transposed convolutions to progressively upsample the reconstructed latent tensor $\hat{Z} \in \mathbb{R}^{C\times T}$ back to the original

sample rate. The output is a reconstructed waveform $\hat{x} \in \mathbb{R}^{1\times L}$, where L matches the input length. The decoder uses the same channel widths, number of residual blocks, and Snake activation [107] as the encoder, following the SEANet design.

3.1.2 Loss Functions

The model is trained using a combination of objectives applied to both the intermediate latent space and the final synthesized waveform. To stabilize training and ensure the encoder latents remain suitable for splatting, we apply a commitment loss inside the bottleneck, defined as $\mathcal{L}_{\text{commit}} = \|Z - \text{sg}(\hat{Z})\|_2^2$, where $\text{sg}(\cdot)$ denotes the stop-gradient operation. This pulls the encoder output toward the splatted reconstruction it has to match. Outer gradients from $\mathcal{L}$ reach the encoder E via a straight-through estimator: the decoder receives $\hat{Z}_{\text{STE}} = Z + \text{sg}(\hat{Z} - Z)$, which has the value of the splat $\hat{Z}$ but carries the gradient of the encoder output Z.

The synthesized waveform $\hat{x}$ is supervised using an outer reconstruction loss comprising an L1 time-domain term $\mathcal{L}_{\text{L1}}$, a multi-scale spectrogram term $\mathcal{L}_{\text{msspec}}$, an adversarial loss $\mathcal{L}_{\text{adv}}$, and a feature-matching loss $\mathcal{L}_{\text{feat}}$ from a multi-scale Short-Time Fourier Transform (STFT) discriminator. The total training loss combines all objectives:

$$\mathcal{L} = \lambda_1 \mathcal{L}_{\text{L1}} + \lambda_s \mathcal{L}_{\text{msspec}} + \lambda_a \mathcal{L}_{\text{adv}} + \lambda_f \mathcal{L}_{\text{feat}} + \lambda_c \mathcal{L}_{\text{commit}}, \tag{3}$$

where $\lambda_1, \lambda_s, \lambda_a, \lambda_f, \lambda_c \in \mathbb{R}^+$ are scalar weighting hyperparameters (concrete values in Appendix A). The gradients from $\mathcal{L}$ are backpropagated through D and E via a gradient balancer.

3.2 Inference

During inference, the model compresses audio segments into a set of quantized parameters, which are then transmitted and decoded to reconstruct the waveform.

3.2.1 Parameter Quantization and Token Format

To achieve compression, each stored parameter of the fitted Gaussians is scalar-quantized to B bits for transmission, where B is the bit depth. The quantization levels are determined via k-means clustering on a held-out calibration subset after training is complete: for each parameter type (scales σ_i and amplitudes $w_{i,c}$ separately), we fit 2^B centroids to the empirical distribution of full-precision values collected from the calibration set. At inference time, each parameter is mapped to its nearest centroid index for transmission, and the receiver looks up the centroid value to dequantize.

Positions are handled in one of two ways. In the frozen-position variant, the centers lie on a uniform temporal grid, $\mu_i = (i-1)(T-1)/(N_G-1)$ for $i \in \{1, \ldots, N_G\}$ and $N_G > 1$; each μ_i is a fixed scalar that does not vary with the temporal index t, and no positions are transmitted. In the learned-position variant, positions are optimized per segment and transmitted as additional parameters. Section 4.2.3 compares the two. With frozen positions, the transmitted parameters per segment consist of N_G scale values σ_i and $N_G \times C$ amplitude values $w_{i,c}$, each quantized to B bits, giving a theoretical bitrate of

$$\text{BR}_{\text{kbps}} = \frac{(C+1)N_G B}{T_{\text{seg}} \times 1000}, \tag{4}$$

where T_{seg} is the duration of the audio segment in seconds. Both N_G and B are integers, so the supported rates are discrete. When positions are learned and transmitted, each μ_i is quantized independently to B_μ bits (a separate budget from B), and the bitrate becomes $\big((C+1)N_G B + N_G B_\mu\big)/(T_{\text{seg}} \times 1000)$.

3.2.2 GS Predictor Net

The inner optimization loop described above requires K gradient steps per segment at encoding time, which is substantially slower than a feed-forward codebook lookup. To address this, we train a GS Predictor Net P that amortises the inner loop, regressing the Gaussian parameters from the continuous latent Z in a single forward pass.

Architecture. The predictor takes the encoder latent $Z \in \mathbb{R}^{C\times T}$ as input and outputs the full parameter set: N_G scale values σ_i, the $N_G \times C$ amplitude matrix $w_{i,c}$, and N_G position values μ_i. A stack of dilated 1D convolutional blocks first extracts temporal features from Z. A transformer decoder then cross-attends from N_G learned queries, one per primitive into these features, producing a per-primitive representation. Three linear output heads map each query to its scale (Softplus), amplitude vector, and position (Sigmoid scaled to $[0, T-1]$). The predictor runs with predicted or frozen centers; in the frozen case the position head is bypassed. Its queries can also be generated from a shared learned token and positional encodings of an N_G-point grid, so one checkpoint serves different N_G.

Training. The predictor is trained after the main encoder-decoder training is complete, with the encoder and decoder weights frozen. The objective is a simple MSE loss between the rendered splat produced from the predicted parameters and the encoder latent:

$$\mathcal{L}_{\text{pred}} = \|\hat{Z}_{\text{splat}} - Z\|_2^2, \tag{5}$$

where $\hat{Z}_{\text{splat}}$ is obtained by evaluating the rendered approximation with the predicted parameters. The position head uses a higher learning rate than the remaining parameters to accelerate convergence of the temporal placement.

Inference. At inference time, the predictor replaces the inner loop entirely: the encoder produces Z, the predictor outputs the primitive parameters in a single pass, and these are scalar-quantized and transmitted. The decoder side is unchanged. This amortisation reduces the encoding cost from K optimization steps to a single forward pass through the predictor, while the reconstructed waveform quality remains close to that of the full iterative fit.

4 Experiments

4.1 Experimental setup

Datasets. For the main comparison with WavTokenizer-protocol baselines (Table 1), we train GS-Codec on Common Voice [2] (≈1,717 h), LibriTTS [98] (≈554 h), and VCTK [91] (≈68 h), all resampled to 24 kHz monaural (≈2,339 h total). Speech reconstruction is evaluated on LibriTTS `test-clean` (clean conditions), LibriTTS `test-other` (noisy conditions), and LJSpeech (out-of-domain generalisation). For Table 2, we train on the 54,000-hour Librilight-Large dataset [33], resampled to 16 kHz. We evaluate on LibriSpeech `test-clean` [62]. For the ablation studies (§4.2.2), all models are trained on the same 2,339-hour mixture and evaluated on LibriSpeech `test-clean`.

Baselines. For Table 1 we compare against EnCodec [16], DAC [42], SpeechTokenizer [101], HiFi-Codec [93], Vocos [72], and WavTokenizer [31], following the evaluation protocol of Ji et al. [31]. Baseline numbers are quoted directly from Ji et al. [31]. For Table 2 we additionally compare against Mimi [17] and XYTokenizer [24].

Metrics. We report UTMOS [69] as the primary perceptual quality metric, alongside PESQ [67], STOI [75], and V/UV F1. Wideband PESQ (WB) is used for the WavTokenizer comparison (Table 1); narrowband PESQ (NB) for Table 2. For Table 2 we additionally report VALL-E speaker similarity (SIM) [85] and WER, computed from transcriptions by `facebook/hubert-large-ls960-ft` [28] on the 4-10 s subset (n=1,237).

Hyperparameters. The SEANet encoder/decoder uses C=32 latent channels, 64 base filters, and Snake activations [107], producing 225 latent frames per 3-second segment at 75 Hz. We train with Adam (β_1=0.5, β_2=0.9, LR 10^{-4}) and EMA (decay 0.99) for 200 epochs of 2,000 steps on a single NVIDIA RTX PRO 6000 GPU (≈144 h). The GS bottleneck uses N_G=80 primitives per 3-second segment and K=100 inner Adam steps during training. At evaluation time we run 300 steps under a cosine schedule. Positions are fixed to a uniform grid; parameters are scalar-quantised with per-type k-means codebooks (2^B centroids) calibrated on `libritts-dev-clean`. We report three decode-time variants for each (N_G, B) configuration: **unfrozen** - positions are optimised per-utterance and transmitted at a fixed B_μ=10 bits, yielding a slight bitrate increase; **frozen** - positions are fixed to a uniform grid and not transmitted (default); and **Pred** - the GS Predictor Network (§3.2.2) replaces

iterative optimisation at encoding time. The two iterative variants serve as upper bounds; **Pred** is the deployable single-pass codec.

4.2 Results

4.2.1 Comparison with open-source codecs

Table 1 presents reconstruction results on LibriTTS `test-clean`, `test-other`, and LJSpeech. At the high-bitrate operating point (N_G=90, 6b, unfrozen), GS-Codec surpasses all comparable-bitrate baselines on UTMOS, PESQ, and STOI, approaching the DAC 9 kbps reference. The trend holds across all three evaluation sets and degrades gracefully at lower rates (N_G=60, 4b, unfrozen, 2.84 kbps still leads multi-codebook baselines at 3 kbps on UTMOS). WavTokenizer remains stronger below 1 kbps, outside our operating range.

Table 2 compares GS-Codec trained on the larger Librilight-Large corpus (54k h) against additional baselines on LibriSpeech `test-clean`, adding WER and SIM. At 6.05 kbps GS-Codec leads on UTMOS (4.01) and SIM (0.94), while DAC retains a NB-PESQ advantage (3.77 vs. 3.71).

Table 3 compares GS, FSQ, and RVQ using the same 32-channel, 64-filter SEANet backbone and 1,237-utterance LibriSpeech `test-clean` subset. GS achieves higher UTMOS and SIM, with comparable PESQ-NB, STOI, and WER at the reported precision. Appendix C gives the training and evaluation settings.

Human listening evaluation. Blinded MUSHRA and MOS studies at approximately 3 and 6 kbps compare GS, FSQ, and RVQ, with DAC and EnCodec as external reference points. At approximately 6 kbps GS obtains higher mean MUSHRA scores and smaller mean MOS gaps to natural speech than FSQ and RVQ; at approximately 3 kbps it obtains smaller mean MOS gaps at similar MUSHRA scores. Appendix D gives the protocol and all scores. Audio samples are available at `https://ronaluf.github.io/gs-codec/`.

4.2.2 Bitrate sweep

Figure 3 sweeps $N_G \in \{50, \ldots, 140\}$ at fixed 5-bit depth (2.75-7.70 kbps). All three metrics improve steeply up to N_G=80 and saturate thereafter. Figure 4 sweeps bit-depth $B \in \{3, \ldots, 10\}$ for N_G=60 and N_G=80; metrics saturate past 6 bits while N_G=80 dominates at every bit-depth.

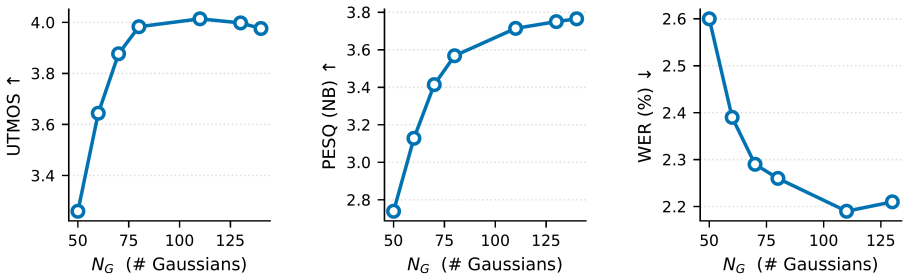


Figure 3: N_G sweep at fixed 5-bit PTQ on LibriSpeech `test-clean` (4-10 s subset, n=1237; WER: `hubert-large-ls960-ft`).

4.2.3 Ablations

We ablate the design choice of fixing primitive positions to a uniform temporal grid versus allowing them to be optimized freely during the inner loop. This choice represents a critical bitrate optimization, as the additional bits required to transmit learned positions do not translate to large perceptual gains at sufficient primitive counts. Figure 5 reports absolute quality and the metric delta between the two conditions across all swept values of N_G at fixed 6-bit post-training quantization.

4.3 Encoding time and throughput

Table 4 reports per-segment GPU execution time on an NVIDIA L40S GPU (200 segments $\times$ 3 s, 10 warmup segments, measured with `torch.cuda.Event`). The iterative GS encoder (300-step Adam,

Table 1: Objective reconstruction results on LibriTTS `test-clean` (clean), LibriTTS `test-other` (noisy), and LJSpeech (out-of-domain), following the protocol of Ji et al. [31] Table 1. PESQ is wideband (WB). Best results in each band are **bold**.

Dataset	Band	Model	BW ↓	tok/s ↓	UTMOS ↑	PESQ ↑	STOI ↑	V/UV F1 ↑
LibriTTS test-clean		DAC (ref)	9.0 kbps	900	3.910	3.908	0.970	0.978
	≥5 kbps high	EnCodec	6.0 kbps	600	3.040	2.720	0.939	0.953
		Vocos	6.0 kbps	600	3.695	2.807	0.943	0.944
		SpeechTokenizer	6.0 kbps	600	3.879	2.612	0.917	0.950
		GS-Codec (N_G=90, 6b, Pred, frozen)	5.94 kbps	990	4.076	3.289	0.970	0.956
		GS-Codec (N_G=90, 6b, frozen)	5.94 kbps	990	4.094	3.348	0.972	**0.960**
		GS-Codec (N_G=90, 6b, unfrozen)	6.24 kbps	1020	**4.102**	**3.400**	**0.974**	0.958
	3-5 kbps mid	DAC	4.0 kbps	400	3.433	2.738	0.928	0.957
		HiFi-Codec	3.0 kbps	400	3.753	2.961	0.941	**0.962**
		HiFi-Codec	4.0 kbps	300	3.904	3.012	0.945	0.958
		EnCodec	3.0 kbps	300	2.307	2.052	0.901	0.920
		Vocos	3.0 kbps	300	3.539	2.403	0.923	0.936
		SpeechTokenizer	3.0 kbps	300	3.563	1.931	0.878	0.927
		GS-Codec (N_G=80, 4b, Pred, frozen)	3.52 kbps	880	3.990	3.071	0.959	0.943
		GS-Codec (N_G=80, 4b, frozen)	3.52 kbps	880	3.974	3.067	0.960	0.950
		GS-Codec (N_G=80, 4b, unfrozen)	3.79 kbps	907	**4.026**	**3.177**	**0.966**	0.953
	<3 kbps low	DAC	1.0 kbps	100	1.494	1.246	0.771	0.794
		WavTokenizer	0.5 kbps	40	3.602	1.703	0.862	0.917
		WavTokenizer	0.9 kbps	75	**4.049**	2.373	0.914	0.938
		GS-Codec (N_G=50, 4b, unfrozen)	2.37 kbps	567	3.521	2.409	0.932	0.930
		GS-Codec (N_G=60, 4b, frozen)	2.64 kbps	660	3.568	2.453	0.929	0.933
		GS-Codec (N_G=60, 4b, unfrozen)	2.84 kbps	680	3.847	**2.825**	**0.951**	**0.940**
LibriTTS test-other		DAC (ref)	9.0 kbps	900	3.357	3.760	0.958	0.970
	≥5 kbps high	EnCodec	6.0 kbps	600	2.657	2.682	0.924	0.934
		Vocos	6.0 kbps	600	3.196	2.559	0.921	0.920
		SpeechTokenizer	6.0 kbps	600	3.285	2.327	0.881	0.921
		GS-Codec (N_G=90, 6b, Pred, frozen)	5.94 kbps	990	3.623	3.011	0.955	0.939
		GS-Codec (N_G=90, 6b, frozen)	5.94 kbps	990	3.631	3.080	0.958	0.943
		GS-Codec (N_G=90, 6b, unfrozen)	6.24 kbps	1020	**3.639**	**3.131**	**0.960**	**0.943**
	3-5 kbps mid	DAC	4.0 kbps	400	2.945	2.595	0.908	0.940
		HiFi-Codec	4.0 kbps	400	3.075	2.554	0.913	0.939
		HiFi-Codec	3.0 kbps	300	3.303	2.608	0.917	0.932
		EnCodec	3.0 kbps	300	2.088	2.052	0.884	0.893
		Vocos	3.0 kbps	300	3.056	2.193	0.897	0.905
		SpeechTokenizer	3.0 kbps	300	3.018	1.737	0.837	0.891
		GS-Codec (N_G=80, 4b, Pred, frozen)	3.52 kbps	880	3.455	2.786	0.938	0.932
		GS-Codec (N_G=80, 4b, frozen)	3.52 kbps	880	3.504	2.831	0.944	0.934
		GS-Codec (N_G=80, 4b, unfrozen)	3.79 kbps	907	**3.539**	**2.907**	**0.950**	**0.942**
	<3 kbps low	DAC	1.0 kbps	100	1.499	1.245	0.751	0.778
		WavTokenizer	0.5 kbps	40	3.055	1.662	0.834	0.895
		WavTokenizer	0.9 kbps	75	**3.431**	2.261	0.891	0.917
		GS-Codec (N_G=50, 4b, unfrozen)	2.37 kbps	567	3.117	2.310	0.915	0.926
		GS-Codec (N_G=60, 4b, frozen)	2.64 kbps	660	3.130	2.331	0.912	0.926
		GS-Codec (N_G=60, 4b, unfrozen)	2.84 kbps	680	3.378	**2.620**	**0.934**	**0.934**
LJSpeech		DAC (ref)	9.0 kbps	900	4.301	3.902	0.973	0.976
	≥5 kbps high	EnCodec	6.0 kbps	600	3.229	2.663	0.944	**0.956**
		Vocos	6.0 kbps	600	4.033	2.926	0.950	0.946
		SpeechTokenizer	6.0 kbps	600	4.237	2.641	0.932	0.945
		GS-Codec (N_G=90, 6b, Pred, frozen)	5.94 kbps	990	4.346	3.205	0.970	0.952
		GS-Codec (N_G=90, 6b, frozen)	5.94 kbps	990	4.347	3.273	0.971	0.944
		GS-Codec (N_G=90, 6b, unfrozen)	6.24 kbps	1020	**4.372**	**3.356**	**0.975**	0.944
	3-5 kbps mid	DAC	4.0 kbps	400	3.811	2.762	0.934	**0.952**
		HiFi-Codec	4.0 kbps	400	4.166	2.763	0.945	0.950
		HiFi-Codec	3.0 kbps	300	4.269	2.909	0.949	0.947
		EnCodec	3.0 kbps	300	2.391	2.019	0.906	0.933
		Vocos	3.0 kbps	300	3.788	2.501	0.931	0.939
		SpeechTokenizer	3.0 kbps	300	3.991	2.046	0.902	0.930
		GS-Codec (N_G=80, 4b, frozen)	3.52 kbps	880	4.256	2.998	0.960	0.942
		GS-Codec (N_G=80, 4b, unfrozen)	3.79 kbps	907	**4.338**	**3.167**	**0.968**	0.944
	<3 kbps low	DAC	1.0 kbps	100	1.444	1.208	0.782	0.810
		WavTokenizer	0.5 kbps	40	4.019	2.114	0.909	0.941
		WavTokenizer	0.9 kbps	75	**4.258**	2.492	0.931	0.940
		GS-Codec (N_G=60, 4b, frozen)	2.64 kbps	660	3.602	2.312	0.925	0.929
		GS-Codec (N_G=60, 4b, unfrozen)	2.84 kbps	680	4.046	**2.666**	**0.950**	**0.941**

Table 2: Comparison with open-source neural audio codecs on LibriSpeech `test-clean`. WER evaluated with `facebook/hubert-large-ls960-ft` on 4-10 s subset. Acoustic metrics on RVQ1:8 for stacked codecs; for GS-Codec this is the full 33-row token stream (see Eq. 4). Baseline results from published results.

	Model	BR (kbps)	Param	WER↓ (%)	PESQ↑	UTMOS↑	SIM↑
≥5 kbps high	DAC-75Hz	6.0	74M	2.27	**3.77**	3.62	0.90
	EnCodec-75Hz	6.0	15M	2.24	3.12	3.01	0.89
	GS-Codec (130g, 4b, frozen)	5.72	55M	2.25	3.67	3.93	0.93
	GS-Codec (120g, 4b, frozen)	5.28	55M	2.22	3.65	3.94	0.93
	GS-Codec (110g, 5b, frozen)	6.05	55M	**2.19**	3.71	**4.01**	**0.94**
3-5 kbps mid	SpeechTokenizer-50Hz	4.0	103M	2.47	3.01	3.90	0.85
	GS-Codec (70g, 5b, frozen)	3.85	55M	2.29	3.41	3.88	0.91
	GS-Codec (80g, 4b, frozen)	3.52	55M	2.27	3.46	3.91	0.91
	GS-Codec (110g, 4b, frozen)	4.84	55M	**2.26**	**3.63**	**3.95**	**0.93**
<3 kbps low	Mimi-12.5Hz	1.1	78M	3.15	2.75	3.56	0.73
	XYTokenizer-12.5Hz	1.0	520M	**2.36**	3.00	**4.00**	0.84
	GS-Codec (60g, 4b, unfrozen)	2.84	55M	2.43	**3.02**	3.54	**0.87**

Table 3: Same-backbone comparison at approximately 6 kbps. Values are means ± 95% utterance-bootstrap confidence interval half-widths ($n = 1{,}237$). PESQ is narrowband. GS uses predicted centers.

	RVQ	FSQ	GS Predictor
Payload (kbps)	6.000	6.000	5.950
PESQ-NB ↑	$3.8 \pm .011$	$3.8 \pm .013$	$3.8 \pm .009$
UTMOS ↑	$3.94 \pm .020$	$3.94 \pm .019$	$4.05 \pm .017$
STOI ↑	$.96 \pm .0008$	$.96 \pm .0008$	$.96 \pm .0011$
SIM ↑	$.93 \pm .0013$	$.92 \pm .0014$	$.94 \pm .0012$
WER (%) ↓	$2.1 \pm .25$	$2.2 \pm .26$	$2.1 \pm .25$

paper evaluation config) takes 171 ms per segment (RTF = 0.057), roughly $34\times$ slower than DAC and $32\times$ slower than EnCodec. The GS Predictor Net reduces this to 7.9 ms (RTF = 0.003), competitive with DAC ($1.6\times$) and EnCodec ($1.5\times$). Measurements exclude the decoder; the timed predictor uses frozen centers.

Table 4: Encoder GPU execution time on NVIDIA L40S. RTF $= t_{\text{encode}}/t_{\text{audio}}$; lower is better. Measured over 200 segments × 3 s with 10 warmup segments. `torch.cuda.Event` timing; single run.

Model	Mean	Median	p95	RTF
GS-Codec iterative (300-step Adam)	171.2 ms	152.6 ms	208.4 ms	0.057
GS-Pred	7.9 ms	7.9 ms	7.9 ms	0.0026
EnCodec 24 kHz @ 6 kbps	5.4 ms	5.3 ms	5.4 ms	0.0018
DAC 24 kHz @ 6 kbps	5.0 ms	5.0 ms	5.1 ms	0.0017

5 Conclusion

We introduced GS-Codec, a neural audio codec that replaces conventional quantization bottlenecks with a parametric one-dimensional Gaussian-splatting decomposition. Instead of relying on residual vector quantization or finite scalar quantization, GS-Codec represents encoder latents as a weighted sum of Gaussian primitives, enabling compression through structured signal decomposition.

Experimental results demonstrate that GS-Codec achieves competitive or superior perceptual quality, intelligibility, and speaker similarity compared to established open-source codecs across multiple bitrate regimes. In addition, the proposed GS Predictor Net amortizes the iterative fitting process into a single forward pass, substantially reducing encoding time while maintaining reconstruction quality close to the full optimization-based approach.

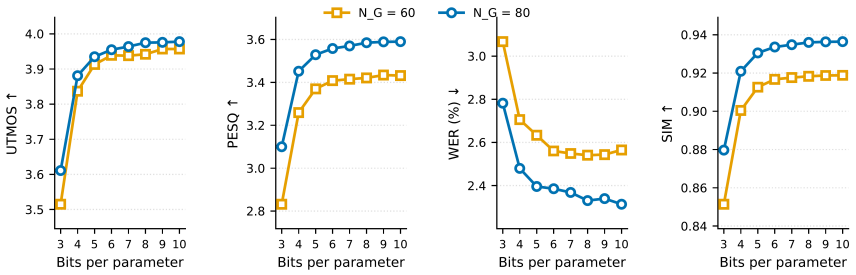


Figure 4: Bit-depth sweep for N_G=60 (orange squares) and N_G=80 (blue circles) on LibriSpeech `test-clean` (4-10 s subset, n=1237). Metrics saturate past 6 bits; N_G=80 outperforms N_G=60 at every bit-depth.

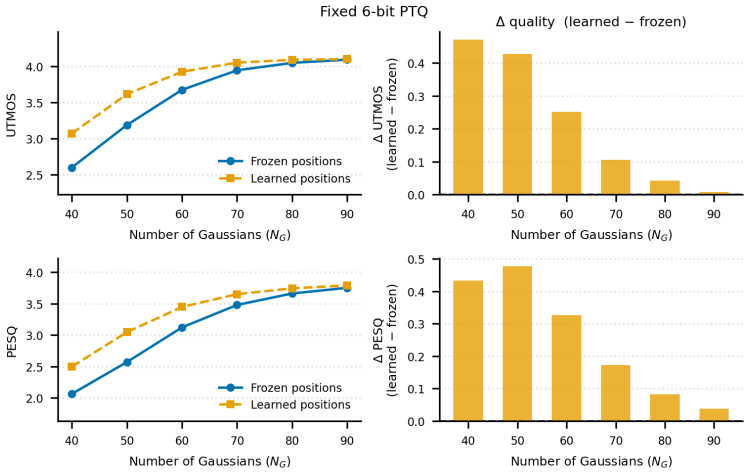


Figure 5: Effect of unfreezing primitive positions at fixed 6-bit PTQ. Left column: absolute UTMOS and PESQ versus N_G for frozen (solid blue) and learned (dashed orange) positions. Right column: metric delta Δ = learned − frozen.

Overall, our results suggest that parametric latent decompositions provide a promising alternative to partition-based neural audio compression and open new directions for fine-grained post-training rate control in neural codecs.

Acknowledgments and Disclosure of Funding

This work was supported by the D.E. Koshland Jr. Family Career Development Chair in Advanced Technologies in Electrical and Computer Engineering. The authors declare no competing interests.

References

[1] Andrea Agostinelli, Timo I. Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Caillon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, Matt Sharifi, Neil Zeghidour, and Christian Frank. MusicLM: Generating music from text. *arXiv preprint arXiv:2301.11325*, 2023.

[2] Rosana Ardila, Megan Branson, Kelly Davis, Michael Henretty, Michael Kohler, Josh Meyer, Reuben Morais, Lindsay Saunders, Francis M. Tyers, and Gregor Weber. Common voice: A massively-multilingual speech corpus. In *Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020)*, pages 4211–4215, 2020.

[3] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. In *Advances in Neural Information Processing Systems*, volume 33, pages 12449–12460, 2020.

[4] Alexei Baevski, Wei-Ning Hsu, Qiantong Xu, Arun Babu, Jiatao Gu, and Michael Auli. data2vec: A general framework for self-supervised learning in speech, vision and language. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 1298–1312. PMLR, 2022. URL `https://proceedings.mlr.press/v162/baevski22a.html`.

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

[6] Zalán Borsos, Raphaël Marinier, Damien Vincent, Eugene Kharitonov, Olivier Pietquin, Matt Sharifi, Dominik Roblek, Olivier Teboul, David Grangier, Marco Tagliasacchi, et al. Audiolm: a language modeling approach to audio generation. *IEEE/ACM transactions on audio, speech, and language processing*, 31:2523–2533, 2023.

[7] Zalán Borsos, Matt Sharifi, Damien Vincent, Eugene Kharitonov, Neil Zeghidour, and Marco Tagliasacchi. Soundstorm: Efficient parallel audio generation. *arXiv preprint arXiv:2305.09636*, 2023.

[8] Tom B. Brown, Benjamin Mann, Nick Ryder, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, 2020. Best Paper Award.

[9] Yunkee Chae, Woosung Choi, Yuhta Takida, Junghyun Koo, Yukara Ikemiya, Zhi Zhong, Kin Wai Cheuk, Marco A Martínez-Ramírez, Kyogu Lee, Wei-Hsiang Liao, et al. Variable bitrate residual vector quantization for audio coding. pages 1–5, 2025.

[10] David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19457–19467, 2024.

[11] Sanyuan Chen, Chengyi Wang, Zhengyang Chen, Yu Wu, Shujie Liu, Zhuo Chen, Jinyu Li, Naoyuki Kanda, Takuya Yoshioka, Xiong Xiao, et al. Wavlm: Large-scale self-supervised pre-training for full stack speech processing. *IEEE Journal of Selected Topics in Signal Processing*, 16(6):1505–1518, 2022.

[12] Sanyuan Chen, Shujie Liu, Long Zhou, Yanqing Liu, Xu Tan, Jinyu Li, Sheng Zhao, Yao Qian, and Furu Wei. VALL-E 2: Neural codec language models are human parity zero-shot text to speech synthesizers. *arXiv preprint arXiv:2406.05370*, 2024.

[13] Yuhan Chen, Wenxuan Yu, Guofa Li, Yijun Xu, Ying Fang, Yicui Shi, Long Cao, Wenbo Chu, and Keqiang Li. LL-GaussianImage: Efficient image representation for zero-shot low-light enhancement with 2D Gaussian splatting. *arXiv preprint arXiv:2601.15772*, 2026.

[14] Yu-An Chung, Yu Zhang, Wei Han, Chung-Cheng Chiu, James Qin, Ruoming Pang, and Yonghui Wu. W2v-bert: Combining contrastive learning and masked language modeling for self-supervised speech pre-training. In *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 244–250. IEEE, 2021.

[15] Jade Copet, Felix Kreuk, Itai Gat, Tal Remez, David Kant, Gabriel Synnaeve, Yossi Adi, and Alexandre Défossez. Simple and controllable music generation. In *Advances in Neural Information Processing Systems*, volume 36, pages 47704–47720, 2023.

[16] Alexandre Défossez, Jade Copet, Gabriel Synnaeve, and Yossi Adi. High fidelity neural audio compression. *Transactions on Machine Learning Research (TMLR)*, 2023.

[17] Alexandre Défossez, Laurent Mazaré, Manu Orsini, Amélie Royer, Patrick Pérez, Hervé Jégou, Edouard Grave, and Neil Zeghidour. Moshi: A speech-text foundation model for real-time dialogue. *arXiv preprint arXiv:2410.00037*, 2024.

[18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.

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

[20] Zhihao Du, Qian Chen, Shiliang Zhang, Kai Hu, Heng Lu, Yexin Yang, Hangrui Hu, Siqi Zheng, Yue Gu, Ziyang Ma, Zhifu Gao, and Zhijie Yan. CosyVoice: A scalable multilingual zero-shot text-to-speech synthesizer based on supervised semantic tokens. *arXiv preprint arXiv:2407.05407*, 2024.

[21] Emilien Dupont, Hyunjik Kim, SM Ali Eslami, Danilo Jimenez Rezende, and Dan Rosenbaum. From data to functa: Your data point is a function and you can treat it like one. In *International Conference on Machine Learning*, pages 5694–5725. PMLR, 2022.

[22] Emilien Dupont, Hrushikesh Loya, Milad Alizadeh, Adam Golinski, Yee Whye Teh, and Arnaud Doucet. Coin++: Neural compression across modalities. *Transactions on Machine Learning Research*, 2022.

[23] Jesse Engel, Lamtharn Hantrakul, Chenjie Gu, and Adam Roberts. DDSP: Differentiable digital signal processing. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=B1x1ma4tDr`.

[24] Yitian Gong, Luozhijie Jin, Ruifan Deng, Dong Zhang, Xin Zhang, Qinyuan Cheng, Zhaoye Fei, Shimin Li, and Xipeng Qiu. XY-Tokenizer: Mitigating the semantic-acoustic conflict in low-bitrate speech codecs, 2025. URL `https://arxiv.org/abs/2506.23325`.

[25] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, volume 27, 2014.

[26] Robert M. Gray. Vector quantization. *IEEE ASSP Magazine*, 1(2):4–29, 1984.

[27] Ben Hayes, Jordie Shier, George Fazekas, Andrew McPherson, and Charalampos Saitis. A review of differentiable digital signal processing for music and speech synthesis. *Frontiers in Signal Processing*, 3, 2023. doi: 10.3389/frsip.2023.1284100.

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

[29] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH 2024 conference papers*, pages 1–11, 2024.

[30] Won Jang, Dan Lim, Jaesam Yoon, Bongwan Kim, and Juntae Kim. Univnet: A neural vocoder with multi-resolution spectrogram discriminators for high-fidelity waveform generation. In *Interspeech 2021*, pages 2207–2211, 2021. doi: 10.21437/Interspeech.2021-1100.

[31] Shengpeng Ji, Ziyue Jiang, Wen Wang, Yifu Chen, Minghui Fang, Jialong Zuo, Qian Yang, Xize Cheng, Zehan Wang, Ruiqi Li, et al. Wavtokenizer: an efficient acoustic discrete codec tokenizer for audio language modeling. In *The Thirteenth International Conference on Learning Representations*, 2025.

[32] Zeqian Ju, Yuancheng Wang, Kai Shen, Xu Tan, Detai Xin, Dongchao Yang, Eric Liu, Yichong Leng, Kaitao Song, Siliang Tang, Zhizheng Wu, Tao Qin, Xiangyang Li, Wei Ye, Shikun Zhang, Jiang Bian, Lei He, Jinyu Li, and Sheng Zhao. NaturalSpeech 3: Zero-shot speech synthesis with factorized codec and diffusion models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 22605–22623. PMLR, 2024. URL `https://proceedings.mlr.press/v235/ju24b.html`.

[33] Jacob Kahn, Morgane Rivière, Weiyi Zheng, Evgeny Kharitonov, Qiantong Xu, Pierre-Emmanuel Mazaré, Julien Karadayi, Vitaliy Liptchinsky, Ronan Collobert, Christian Fuegen, Tatiana Likhomanenko, Gabriel Synnaeve, Armand Joulin, Abdelrahman Mohamed, and Emmanuel Dupoux. Libri-Light: A benchmark for ASR with limited or no supervision. In *ICASSP 2020 – 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7669–7673, 2020. doi: 10.1109/ICASSP40776.2020.9052942.

[34] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, George Drettakis, et al. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023.

[35] Eugene Kharitonov et al. Speak, read and prompt: High-fidelity text-to-speech with minimal supervision. *Transactions of the Association for Computational Linguistics*, 2023.

[36] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*, 2015. URL `https://openreview.net/forum?id=8gmWwjFyLj`.

[37] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations*, 2014.

[38] Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. In *Advances in Neural Information Processing Systems*, volume 33, pages 17022–17033, 2020.

[39] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. DiffWave: A versatile diffusion model for audio synthesis. In *9th International Conference on Learning Representations (ICLR 2021)*, 2021. URL https://openreview.net/forum?id=a-xFK8Ymz5J.

[40] Felix Kreuk, Gabriel Synnaeve, Adam Polyak, Uriel Singer, Alexandre Défossez, Jade Copet, Devi Parikh, Yaniv Taigman, and Yossi Adi. Audiogen: Textually guided audio generation. In *The Eleventh International Conference on Learning Representations*, 2023.

[41] Kundan Kumar, Rithesh Kumar, Thibault De Boissiere, Lucas Gestin, Wei Zhen Teoh, Jose Sotelo, Alexandre De Brebisson, Yoshua Bengio, and Aaron C Courville. Melgan: Generative adversarial networks for conditional waveform synthesis. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[42] Rithesh Kumar, Prem Seetharaman, Alejandro Luebs, Ishaan Kumar, and Kundan Kumar. High-fidelity audio compression with improved rvqgan. In *Advances in Neural Information Processing Systems*, volume 36, pages 27980–27993, 2023.

[43] Kushal Lakhotia, Eugene Kharitonov, Wei-Ning Hsu, Yossi Adi, Adam Polyak, Benjamin Bolte, Tu-Anh Nguyen, Jade Copet, Alexei Baevski, Abdelrahman Mohamed, and Emmanuel Dupoux. On generative spoken language modeling from raw audio. *Transactions of the Association for Computational Linguistics*, 9:1336–1354, 2021. doi: 10.1162/tacl_a_00430. URL https://aclanthology.org/2021.tacl-1.79/.

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

[45] Luca A. Lanzendörfer and Roger Wattenhofer. Siamese SIREN: Audio compression with implicit neural representations. In *ICML 2023 Workshop on Neural Compression*, 2023. URL https://openreview.net/forum?id=AgkMFYcOmM.

[46] Luca A. Lanzendörfer, Florian Grötschla, Michael Ungerböck, and Roger Wattenhofer. High-fidelity music vocoder using neural audio codecs. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2025.

[47] Matthew Le, Apoorv Vyas, Bowen Shi, Brian Karrer, Leda Sari, Rashel Moritz, Mary Williamson, Vimal Manohar, Yossi Adi, Jay Mahadeokar, et al. Voicebox: Text-guided multilingual universal speech generation at scale. In *Advances in Neural Information Processing Systems*, volume 36, pages 14005–14034, 2023.

[48] Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11523–11532, 2022.

[49] Sang Gil Lee, Wei Ping, Boris Ginsburg, Bryan Catanzaro, and Sungroh Yoon. Bigvgan: A universal neural vocoder with large-scale training. In *11th International Conference on Learning Representations, ICLR 2023*, 2023.

[50] Jiaqi Li, Xiaolong Lin, Zhekai Li, Shixi Huang, Yuancheng Wang, Chaoren Wang, Zhenpeng Zhan, and Zhizheng Wu. DualCodec: A low-frame-rate, semantically-enhanced neural audio codec for speech generation. In *Proc. Interspeech 2025*, pages 4883–4887, 2025.

[51] Jiaqi Li, Yao Qian, Yuxuan Hu, Leying Zhang, Xiaofei Wang, Heng Lu, Manthan Thakker, Jinyu Li, Sheng Zhao, and Zhizheng Wu. Flexicodec: A dynamic neural audio codec for low frame rates. *arXiv preprint arXiv:2510.00981*, 2025.

[52] Tiantian Li, Xinjie Zhang, Xingtong Ge, Tongda Xu, Dailan He, Jun Zhang, and Yan Wang. Gaussianimage++: Boosted image representation and compression with 2d gaussian splatting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 6442–6449, 2026.

[53] Haohe Liu, Xuenan Xu, Yi Yuan, Mengyue Wu, Wenwu Wang, and Mark D. Plumbley. SemantiCodec: An ultra low bitrate semantic audio codec for general sound. *IEEE Journal of Selected Topics in Signal Processing*, 2024.

[54] Shicong Liu, Hongtao Lu, and Junru Shao. Improved residual vector quantization for high-dimensional approximate nearest neighbor search. *arXiv preprint arXiv:1509.05195*, 2015.

[55] Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2): 129–137, 1982. doi: 10.1109/TIT.1982.1056489.

[56] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=Bkg6RiCqY7`.

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

[58] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[59] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, 41(4):102:1–102:15, 2022. doi: 10.1145/3528223.3530127.

[60] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[61] OpenAI. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. doi: 10.48550/arXiv.2303.08774. URL `https://arxiv.org/abs/2303.08774`.

[62] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 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, 2015.

[63] Julian D Parker, Anton Smirnov, Jordi Pons, CJ Carr, Zack Zukowski, Zach Evans, and Xubo Liu. Scaling Transformers for low-bitrate high-quality speech coding. In *The Thirteenth International Conference on Learning Representations*, 2025.

[64] Hashiru Pramuditha, Vishagar Arunan, Saeedha Nazar, Vinasirajan Viruthshaan, Sameera Ramasinghe, Simon Lucey, and Ranga Rodrigo. DARB-Splatting: Generalizing splatting with decaying anisotropic radial basis functions. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2026.

[65] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. In *International conference on machine learning*, pages 28492–28518. PMLR, 2023.

[66] Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[67] Antony W. Rix, John G. Beerends, Michael P. Hollier, and Andries P. Hekstra. Perceptual evaluation of speech quality (PESQ): A new method for speech quality assessment of telephone networks and codecs. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 749–752, 2001.

[68] Paul K. Rubenstein, Chulayuth Asawaroengchai, Duc Dung Nguyen, Ankur Bapna, Zalán Borsos, Félix de Chaumont Quitry, Peter Chen, Dalia El Badawy, Wei Han, Eugene Kharitonov, Neil Zeghidour, et al. AudioPaLM: A large language model that can speak and listen. *arXiv preprint arXiv:2306.12925*, 2023.

[69] Takaaki Saeki, Detai Xin, Wataru Nakata, Tomoki Koriyama, Shinnosuke Takamichi, and Hiroshi Saruwatari. UTMOS: UTokyo-SaruLab system for VoiceMOS challenge 2022. In *Proc. Interspeech 2022*, pages 4521–4525, 2022.

[70] Kai Shen, Zeqian Ju, Xu Tan, Eric Liu, Yichong Leng, Lei He, Tao Qin, Sheng Zhao, and Jiang Bian. NaturalSpeech 2: Latent diffusion models are natural and zero-shot speech and singing synthesizers. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=Rc7dAwVL3v`.

[71] Vincent Sitzmann, Julien N.P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, 2020. URL `https://proceedings.neurips.cc/paper/2020/hash/53c04118df112c13a8c34b38343b9c10-Abstract.html`.

[72] Hubert Siuzdak. Vocos: Closing the gap between time-domain and Fourier-based neural vocoders for high-quality audio synthesis. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=vY9nzQmQBw.

[73] Hubert Siuzdak, Florian Grötschla, and Luca A Lanzendörfer. Snac: Multi-scale neural audio codec. *arXiv preprint arXiv:2410.14411*, 2024.

[74] Stanislaw Szymanowicz, Chrisitian Rupprecht, and Andrea Vedaldi. Splatter image: Ultra-fast single-view 3d reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10208–10217, 2024.

[75] Cees H. Taal, Richard C. Hendriks, Richard Heusdens, and Jesper Jensen. A short-time objective intelligibility measure for time-frequency weighted noisy speech. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4214–4217. IEEE, 2010. doi: 10.1109/ICASSP.2010.5495701.

[76] Marco Tagliasacchi, Yunpeng Li, Karolis Misiunas, and Dominik Roblek. SEANet: A multi-modal speech enhancement network. In *Proc. Interspeech 2020*, pages 1126–1130, 2020. doi: 10.21437/Interspeech.2020-1563.

[77] Matthew Tancik, Ben Mildenhall, Terrance Wang, Divi Schmidt, Pratul P Srinivasan, Jonathan T Barron, and Ren Ng. Learned initializations for optimizing coordinate-based neural representations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2846–2855, 2021.

[78] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. doi: 10.48550/arXiv.2302.13971. URL https://arxiv.org/abs/2302.13971.

[79] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023. doi: 10.48550/arXiv.2307.09288. URL https://arxiv.org/abs/2307.09288.

[80] Jean-Marc Valin and Jan Skoglund. Lpcnet: Improving neural speech synthesis through linear prediction. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5891–5895. IEEE, 2019.

[81] Jean-Marc Valin, Koen Vos, and Timothy Terriberry. Definition of the opus audio codec, 2012.

[82] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. WaveNet: A generative model for raw audio. In *9th ISCA Workshop on Speech Synthesis (SSW 9)*, page 125, 2016. URL https://www.isca-archive.org/ssw_2016/vandenoord16_ssw.html.

[83] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[84] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[85] Chengyi Wang, Sanyuan Chen, Yu Wu, Zi-Hua Zhang, Long Zhou, Shujie Liu, Zhuo Chen, Yanqing Liu, Huaming Wang, Jinyu Li, Lei He, Sheng Zhao, and Furu Wei. Neural codec language models are zero-shot text to speech synthesizers. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 33:705–718, 2025. doi: 10.1109/TASLPRO.2025.3530270.

[86] Yixin Wang, Yifan Hu, Peiyuan Liu, Naiqi Li, Dai Tao, and Shu-Tao Xia. Forecasting as rendering: A 2d gaussian splatting framework for time series forecasting. *arXiv preprint arXiv:2603.02220*, 2026.

[87] Yuancheng Wang, Dekun Chen, Xueyao Zhang, Junan Zhang, Jiaqi Li, and Zhizheng Wu. TaDiCodec: Text-aware diffusion speech tokenizer for speech language modeling. In *Advances in Neural Information Processing Systems*, 2025.

[88] Christopher Wewer, Kevin Raj, Eddy Ilg, Bernt Schiele, and Jan Eric Lenssen. latentsplat: Autoencoding variational gaussians for fast generalizable 3d reconstruction. In *European conference on computer vision*, pages 456–473. Springer, 2024.

[89] Haibin Wu, Ho-Lam Chung, Yi-Cheng Lin, Yuan-Kuei Wu, Xuanjun Chen, Yu-Chi Pai, Hsiu-Hsuan Wang, Kai-Wei Chang, Alex Liu, and Hung-yi Lee. Codec-superb: An in-depth analysis of sound codec models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 10330–10348, 2024.

[90] Haibin Wu, Naoyuki Kanda, Sefik Emre Eskimez, and Jinyu Li. TS3-Codec: Transformer-based simple streaming single codec. In *Proc. Interspeech 2025*, pages 604–608, 2025. doi: 10.21437/Interspeech.2025-921.

[91] Junichi Yamagishi, Christophe Veaux, and Kirsten MacDonald. CSTR VCTK Corpus: English multi-speaker corpus for CSTR voice cloning toolkit (version 0.92), 2019.

[92] Ryuichi Yamamoto, Eunwoo Song, and Jae-Min Kim. Parallel wavegan: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6199–6203. IEEE, 2020.

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

[94] Shu-wen Yang, Po-Han Chi, Yung-Sung Chuang, Cheng-I Jeff Lai, Kushal Lakhotia, Yist Y Lin, Andy T Liu, Jiatong Shi, Xuankai Chang, Guan-Ting Lin, et al. SUPERB: Speech processing universal performance benchmark. In *Interspeech 2021*, pages 1194–1198, 2021. doi: 10.21437/Interspeech.2021-1775.

[95] Zhen Ye, Peiwen Sun, Jiahe Lei, Hongzhan Lin, Xu Tan, Zheqi Dai, Qiuqiang Kong, Jianyi Chen, Jiahao Pan, Qifeng Liu, Yike Guo, and Wei Xue. Codec does matter: Exploring the semantic shortcoming of codec for audio language model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 25697–25705, 2025.

[96] H Yeh. Lyra v2: a better faster and more versatile speech codec. Google Open Source Blog, 2022.

[97] Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. SoundStream: An end-to-end neural audio codec. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:1031–1044, 2022.

[98] Heiga Zen, Viet Dang, Rob Clark, Yu Zhang, Ron J. Weiss, Ye Jia, Zhifeng Chen, and Yonghui Wu. LibriTTS: A corpus derived from LibriSpeech for text-to-speech. In *Proceedings of Interspeech 2019*, pages 1526–1530, 2019.

[99] Zhaojie Zeng, Yuesong Wang, Tao Guan, Chao Yang, and Lili Ju. Instant gaussianimage: A generalizable and self-adaptive image representation via 2d gaussian splatting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 27896–27905, 2025.

[100] Jun Zhan et al. AnyGPT: Unified multimodal LLM with discrete sequence modeling. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

[101] Xin Zhang, Dong Zhang, Shimin Li, Yaqian Zhou, and Xipeng Qiu. SpeechTokenizer: Unified speech tokenizer for speech language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=AF9Q8Vip84`.

[102] Xinjie Zhang, Xingtong Ge, Tongda Xu, Dailan He, Yan Wang, Hongwei Qin, Guo Lu, Jing Geng, and Jun Zhang. Gaussianimage: 1000 fps image representation and compression by 2d gaussian splatting. In *European Conference on Computer Vision*, pages 327–345. Springer, 2024.

[103] Yunxiang Zhang, Bingxuan Li, Alexandr Kuznetsov, Akshay Jindal, Stavros Diolatzis, Kenneth Chen, Anton Sochenov, Anton Kaplanyan, and Qi Sun. Image-gs: Content-adaptive image representation via 2d gaussians. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*, pages 1–11, 2025.

[104] Rui-Chen Zheng, Hui-Peng Du, Xiao-Hang Jiang, Yang Ai, and Zhen-Hua Ling. Ervq: Enhanced residual vector quantization with intra-and-inter-codebook optimization for neural audio codecs. *IEEE Transactions on Audio, Speech and Language Processing*, 2025.

[105] Youqiang Zheng, Weiping Tu, Li Xiao, and Xinmeng Xu. Supercodec: A neural speech codec with selective back-projection network. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 566–570. IEEE, 2024.

[106] Yongxin Zhu, Bocheng Li, Yifei Xin, Zhihua Xia, and Linli Xu. Addressing representation collapse in vector quantized models with one linear layer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22968–22977, 2025.

[107] Liu Ziyin, Tilman Hartwig, and Masahito Ueda. Neural networks fail to learn periodic functions and how to fix it. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL `https://proceedings.neurips.cc/paper/2020/hash/1160453108d3e537255e9f7b931f4e90-Abstract.html`.

A Training and evaluation details

SEANet architecture at 24 kHz. The encoder and decoder use the SEANet convolutional architecture [97] with latent dimension C=32, n_{filters}=64 base channels, Snake activation [107], strided ratios $[8, 5, 4, 2]$ (total downsampling factor 320), kernel size 7, residual kernel size 3, 1 residual layer per block, dilation base 2, 2 LSTM layers, weight-norm, and non-causal (bidirectional) convolutions. This yields a 75 Hz latent frame rate and T=225 frames per 3-second segment.

GS bottleneck hyperparameters.

Parameter	Value
Basis type	Gaussian
N_G (training)	80
K inner steps (training)	100
Inner LR (training)	0.4
Inner LR schedule (training)	none
K inner steps (evaluation)	300
Inner LR (evaluation)	0.05
Inner LR schedule (evaluation)	cosine (5-step warmup, final ratio 0.01)
Position LR multiplier	1.0 (training) / 2.0 (evaluation)
Positions	uniform grid $\mu_i = (i-1)(T-1)/(N_G-1)$
Bias	frozen to zero
Scales	shared across channels
Amplitudes	per-channel
Bound parameters	false (explicit clamping at eval: $\sigma \in [0.1, 100]$, $w \in [-3, 3]$)
Input normalisation	disabled

Loss weights and balancer. L1 time-domain: 0.1; multi-scale spectrogram: 2.0; adversarial (MS-STFT discriminator): 4.0; feature matching: 4.0; commitment penalty: 20.0; SI-SNR: 0 (disabled). Gradient balancer: `balance_grads=true`, EMA decay 0.999, per-batch-item normalisation, total norm target 1.0. Discriminator: single MS-STFT discriminator with FFT sizes $[1024, 2048, 512, 256, 128]$ and corresponding hop lengths $[256, 512, 128, 64, 32]$.

Outer optimiser. Adam, β_1=0.5, β_2=0.9, no weight decay, LR 10^{-4}, no LR scheduler. EMA of model weights: decay 0.99, updated every step, stored on GPU. Training dtype: bfloat16 (autocast disabled; bfloat16 used for gradient accumulation stability).

Dataset and training schedule. 200 epochs $\times$ 2,000 gradient steps; 500,000 samples drawn per epoch with replacement, batch size 16, 8 data workers. Segments of 3 seconds; shorter clips accepted if $\geq 0.5\times$ segment duration. Trained on a single NVIDIA RTX PRO 6000 GPU ($\approx$144 hours).

Evaluation commands. WER: `facebook/hubert-large-ls960-ft`, 4-10 s duration filter on LibriSpeech `test-clean` (n=1,237 files). PESQ: narrowband (NB) for Table 2; wideband (WB) for WavTokenizer comparison (Table 1). UTMOS: `sarulab-speech/utmos22_strong`. SIM: WavLM-Large speaker verification cosine similarity [11].

GS Predictor Net training. The predictor is trained with the codec encoder and decoder frozen, using only the splat-reconstruction penalty loss (λ_{penalty}=1.0; all other losses disabled). Architecture: d_{model}=128, 4 attention heads, 4 transformer blocks, cross-attention from N_G learned queries into dilated-conv features of Z. Training: Adam, LR 5×10^{-4}, cosine schedule (1,000-step warmup, min ratio 0.1), batch size 16, 50 epochs $\times$ 2,000 steps, 3-second segments, no refinement iterations (K_{refine}=0), no parameter noise. The predictor is initialised from the codec checkpoint and trained on the same data mixture.

B Per-parameter quantisation ranges

Table 5 reports the per-type quantisation ranges used for the 6-bit k-means PTQ codebooks shipped with the headline checkpoint. Ranges are estimated once on 30 held-out files from `libritts-dev-clean` and stored alongside the checkpoint as `global_ranges_qat.json`.

Table 5: Per-parameter k-means PTQ codebooks (6-bit, $2^6{=}64$ centroids). Ranges estimated on held-out calibration set; loaded at dequantisation.

Parameter	**Min**	**Max**	**Levels** (2^B)
Scale σ_i	0.5	56.25	64
Amplitude $w_{i,c}$	-3.0	3.0	64

C Controlled evaluation

Setup. RVQ, FSQ, and GS share the SEANet encoder/decoder ($C{=}32$, 64 filters, Snake), Librilight-Large, and a 200-epoch $\times$ 2,000-step schedule. Encoder/decoder: 55.45M parameters. Bottleneck state (parameters and buffers): FSQ 32, GS iterative 0, RVQ 532,488, GS Predictor 1,076,998. RVQ: $n_q{=}8 \times 1{,}024$. FSQ: $[4]{\times}16 + [8]{\times}16$, 80 bits/frame, 6.00 kbps. GS: 102 primitives, 5-bit scales and amplitudes, 10-bit centers, $102(33 \times 5 + 10) = 17{,}850$ bits per 3 s, 5.950 kbps. Evaluation: LibriSpeech `test-clean`, 4–10 s, pinned 1,237 utterances, PESQ-NB, WER with `hubert-large-ls960-ft`. Intervals are paired 95% bootstrap half-widths over utterances; pairwise tests are Wilcoxon signed-rank with Holm correction. Token rate counts one token per quantized scalar or codebook index: GS has the lowest temporal rate but $1.93\times$ the token rate of RVQ and $0.48\times$ that of FSQ (Table 6).

Table 6: Unified evaluation on the pinned 1,237 utterances. DAC and EnCodec are public checkpoints rerun with the same protocol.

System	kbps	PESQ-NB ↑	UTMOS ↑	STOI ↑	SIM ↑	WER ↓	Steps/s	Tok./step	Tok./s
GS iterative, 102g/5b	5.95	$3.8 \pm .009$	$4.07 \pm .017$	$.96 \pm .0010$	$.94 \pm .0012$	$2.1 \pm .26$	34	34	1,156
GS Predictor, 102g/5b	5.95	$3.8 \pm .009$	$4.05 \pm .017$	$.96 \pm .0011$	$.94 \pm .0012$	$2.1 \pm .25$	34	34	1,156
FSQ, 32-dim	6.00	$3.8 \pm .013$	$3.94 \pm .019$	$.96 \pm .0008$	$.92 \pm .0014$	$2.2 \pm .26$	75	32	2,400
RVQ, 8 codebooks	6.00	$3.8 \pm .011$	$3.94 \pm .020$	$.96 \pm .0008$	$.93 \pm .0013$	$2.1 \pm .25$	75	8	600
DAC-75Hz	6.00	$3.7 \pm .010$	$3.62 \pm .023$	$.95 \pm .0007$	$.88 \pm .0017$	$2.3 \pm .26$	75	8	600
EnCodec-75Hz	6.00	$3.1 \pm .018$	$3.09 \pm .027$	$.93 \pm .0010$	$.89 \pm .0015$	$2.2 \pm .26$	75	8	600

Subgroup analysis. Same 1,237 utterances; for each condition the extreme tercile ($n{=}412$) by median voiced F0, a ZCR/high-band-energy frication proxy, 95th-percentile spectral flux (transients), fraction of frames below 5% peak RMS (low energy), and peak-to-floor RMS ratio (background level). GS Predictor (102g/5b, 5.950 kbps) against same-backbone FSQ and RVQ (6.000 kbps); paired bootstrap CIs and Wilcoxon tests with Holm correction (Table 7).

Table 7: UTMOS by subgroup.

Subset	GS Predictor	FSQ	RVQ	Δ vs FSQ [95% CI]	Holm p
High F0	3.97	3.84	3.85	$+0.125$ [$+0.109$, $+0.140$]	3×10^{-43}
High frication proxy	4.01	3.88	3.90	$+0.122$ [$+0.108$, $+0.137$]	6×10^{-47}
High spectral flux	4.04	3.95	3.95	$+0.096$ [$+0.083$, $+0.109$]	8×10^{-40}
Low energy	4.05	3.94	3.93	$+0.114$ [$+0.100$, $+0.129$]	7×10^{-46}
Lowest peak-to-floor ratio	3.98	3.86	3.86	$+0.119$ [$+0.106$, $+0.132$]	2×10^{-50}
All	4.05	3.94	3.94	$+0.109$ [$+0.101$, $+0.116$]	2×10^{-133}

Representation ablation. Same trained encoder and utterances, payloads within 0.85% of 17,850 bits, plain PTQ. Basis rows send 32 amplitudes and one width per primitive at 5 bits, plus a 10-bit center when sent; Gabor adds frequency and phase. Removing the decomposition roughly doubles NMSE; changing the atom family changes it by at most 17% (Table 8).

Continuous-autoencoder control. Same backbone; the bottleneck linearly resamples the latent $225 \to 102$ frames and back during training (no quantizer), followed by per-channel k-means PTQ. Single checkpoint trained at 102 frames. At its trained count the AE trails GS by 0.066 UTMOS with fewer bits; away from it, by 0.85–0.88 UTMOS at matched bits (Table 9).

Table 8: Latent NMSE at matched payload.

Representation	Values	Bits / 3 s	NMSE ↓
Gaussian, free centers, 102	3,468	17,850	.0259
Triangular, 102	3,468	17,850	.0275
Laplacian, 102	3,468	17,850	.0295
Gabor, 97	3,492	17,945	.0303
Gaussian, grid centers, 109	3,597	17,985	.0317
No decomposition: decimate to 102 frames	3,264	17,850	.0496
No decomposition: framewise SQ	7,200	18,000	.0537

Table 9: GS versus a continuous autoencoder with post-hoc k-means. UTMOS on the 1,237 utterances.

GS N_G/bits	kbps	UTMOS	AE frames/bits	kbps	UTMOS	Δ
102/5	5.950	4.051	102/5 (trained)	5.440	3.985	$+0.066$
91/5	5.005	3.699	94/5	5.013	2.823	$+0.876$
131/4	5.764	3.969	136/4	5.803	3.118	$+0.851$
127/5	6.985	4.025	131/5	6.987	3.145	$+0.880$

Training cost. Wall-clock time per optimizer update; single L40S, bf16, batch 16, identical backbone, data, and losses (Table 10). The inner fit is batched and its history is not traversed by the outer backward pass.

Table 10: Training cost per update.

Bottleneck	Time/update	Updates/s	400k updates
FSQ	1.160 s	0.862	≈5.3 days
GS, 100-step inner loop	1.209 s	0.827	≈5.6 days
RVQ, 8 codebooks	1.319 s	0.758	≈6.1 days

D Human listening evaluation

Ten LibriSpeech `test-clean` utterances (4.0–5.5 s), one per speaker, selected by a seeded random procedure without inspecting method scores. MOS: each condition rated independently on a 1–5 scale ("Listen to each clip once or twice, then rate its overall quality from 1 to 5"). MUSHRA: blinded 0–100 interface with a hidden reference and a 3.5-kHz anchor, randomized order ("Rate the overall audio quality of each sample relative to the reference"). Participants confirmed headphone use and a quiet room; they were unpaid volunteers and took part anonymously. Ratings are averaged within listener, then across listeners; intervals are participant-level bootstrap. MOS gap is natural speech minus system, paired within listener; lower is better.

At ≈6 kbps all GS configurations obtain higher mean MUSHRA and smaller mean MOS gaps than FSQ and RVQ; at ≈3 kbps they obtain smaller mean MOS gaps at similar MUSHRA (Table 11). The Predictor's MOS gap is smaller at ≈3 than at ≈6 kbps, with overlapping intervals; its MUSHRA ordering is as expected.

Table 11: Listening scores. MUSHRA: mean ± 95% CI half-width. MOS gap: mean [95% CI].

System	kbps	MUSHRA ↑	MOS gap ↓
Hidden reference / natural speech	n/a	78.62 ± 5.41	0.00 [0.00, 0.00]
GS iterative, unfrozen, 100g/5b	5.833	77.67 ± 5.01	0.14 [−0.28, 0.56]
GS Predictor, 109g/5b	5.995	75.36 ± 5.30	0.61 [0.19, 1.03]
GS iterative, frozen, 110g/5b	6.050	75.13 ± 5.14	0.50 [0.00, 0.97]
RVQ	6.000	72.53 ± 5.66	0.69 [0.25, 1.14]
FSQ	3.000	70.78 ± 5.78	0.78 [0.22, 1.31]
RVQ	3.000	70.60 ± 5.51	0.78 [0.36, 1.19]
GS iterative, unfrozen, 63g/4b	2.982	70.60 ± 6.12	0.39 [−0.14, 0.92]
GS iterative, frozen, 68g/4b	2.992	69.49 ± 5.92	0.50 [0.14, 0.86]
GS Predictor, 80g/3b	2.907	68.82 ± 5.86	0.33 [−0.11, 0.78]
FSQ	6.000	67.74 ± 6.01	0.78 [0.31, 1.25]
DAC 24 kHz	6.000	64.22 ± 5.53	0.97 [0.58, 1.36]
EnCodec 24 kHz	6.000	53.68 ± 6.25	1.36 [0.92, 1.78]
3.5-kHz anchor	n/a	36.19 ± 5.74	n/a

E Fixed-budget allocation

Frozen positions; each budget holds $N_G B$ fixed. Intervals are 95% bootstrap half-widths, 10,000 resamples. At 5.28 kbps, 120g/4b beats 160g/3b and 80g/6b on PESQ-NB, UTMOS, and SIM; at N_G=60, raising precision from 5 to 10 bits moves UTMOS only from 2.845 to 2.897 (Table 12). Sufficient primitive capacity comes first; the remaining budget goes to precision.

Table 12: Fixed-budget allocation between primitive count and precision. WER in percent.

kbps	N_G	Bits	PESQ-NB ↑	UTMOS ↑	WER ↓	SIM ↑
3.30	100	3	$2.926 \pm .055$	$3.425 \pm .098$	$2.48 \pm .74$	$.844 \pm .006$
3.30	60	5	$2.484 \pm .042$	$2.845 \pm .066$	$2.38 \pm .72$	$.797 \pm .009$
3.96	120	3	$3.155 \pm .051$	$3.583 \pm .103$	$2.33 \pm .73$	$.867 \pm .007$
3.96	60	6	$2.515 \pm .044$	$2.880 \pm .066$	$2.48 \pm .74$	$.804 \pm .009$
3.96	40	9	$1.898 \pm .031$	$1.905 \pm .065$	5.48 ± 1.35	$.612 \pm .013$
4.40	100	4	$3.494 \pm .044$	$3.889 \pm .087$	$2.02 \pm .68$	$.912 \pm .004$
4.40	80	5	$3.181 \pm .050$	$3.593 \pm .080$	$2.33 \pm .71$	$.895 \pm .005$
4.40	40	10	$1.899 \pm .031$	$1.908 \pm .064$	5.59 ± 1.35	$.612 \pm .013$
5.28	160	3	$3.327 \pm .051$	$3.610 \pm .105$	$2.02 \pm .66$	$.889 \pm .005$
5.28	120	4	$3.685 \pm .038$	$3.953 \pm .086$	$2.02 \pm .66$	$.930 \pm .004$
5.28	80	6	$3.225 \pm .050$	$3.640 \pm .081$	$2.28 \pm .71$	$.901 \pm .005$
5.28	60	8	$2.532 \pm .044$	$2.895 \pm .068$	$2.48 \pm .74$	$.807 \pm .008$
6.60	120	5	$3.805 \pm .035$	$4.039 \pm .080$	$2.17 \pm .68$	$.944 \pm .003$
6.60	100	6	$3.672 \pm .039$	$3.996 \pm .083$	$2.17 \pm .69$	$.933 \pm .004$
6.60	60	10	$2.532 \pm .044$	$2.897 \pm .069$	$2.53 \pm .76$	$.807 \pm .008$
7.92	120	6	$3.836 \pm .034$	$4.061 \pm .078$	$2.12 \pm .67$	$.948 \pm .003$
7.92	80	9	$3.235 \pm .051$	$3.646 \pm .080$	$2.33 \pm .74$	$.903 \pm .005$

F Limitations

GS-Codec has the following limitations. First, encoding is slower than feed-forward codecs: even with the GS Predictor Net, encoding time remains higher than EnCodec and DAC, and the iterative inner loop is substantially slower when the predictor is not used. Second, at lower bitrates (Table 2, <3 kbps band), perceptual quality leaves room for improvement; the baselines in this band operate at different rates. Our experiments evaluate English speech at moderate bitrates. Generalisation to other audio domains and sub-kbps coding remains future work. The current codec processes complete 3-second segments with non-causal components, so GPU execution time does not represent streaming latency.

G Broader impact

GS-Codec can reduce the storage and transmission costs of speech. As with other speech codecs, reconstructed audio can retain speaker identity and private information; use should respect data consent and licenses. Downstream speech-synthesis applications share the potential misuse risks of existing audio representations.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Section 1 describe the Gaussian bottleneck, predictor, and fine-grained post-training rate control. The evaluation concerns speech, and the distinction between codec rate control and a fixed-count predictor is stated in Section 3.2.2.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Appendix F discusses encoding cost, lower-rate perceptual quality, the evaluated audio domain, and segment-based processing.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper is entirely empirical. It contains no theorems, lemmas, or formal proofs. Mathematical formulations (the GS rendering equation, loss functions) are model definitions, not theoretical results requiring proof.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: §4.1 and Appendix A provide the full architecture (SEANet config, ratios, activations), GS bottleneck hyperparameters, loss weights, optimizer settings, and evaluation commands. Training data (CommonVoice, LibriTTS, VCTK) and evaluation data (LibriSpeech `test-clean`) are all publicly available. Audio samples are available at `https://ronaluf.github.io/gs-codec/`; code is provided in the supplementary material.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Audio samples are publicly available at `https://ronaluf.github.io/gs-codec/`; code is provided in the supplementary material. All training data (Common-Voice CC0, LibriTTS CC-BY-4.0, VCTK CC-BY-4.0) and evaluation data (LibriSpeech `test-clean` CC-BY-4.0) are publicly available. Appendix A provides evaluation commands and all hyperparameters needed to run the pipeline.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: §4.1 covers datasets, baselines, metrics, optimizer (Adam, β_1=0.5, β_2=0.9, LR 10^{-4}), training schedule (200 epochs $\times$ 2,000 steps), and key GS hyperparameters (N_G=80, K=100 inner steps). Appendix A provides a complete hyperparameter table, loss weights, discriminator config, and exact evaluation commands.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Table 3 reports 95% utterance-bootstrap confidence intervals, and Appendix C states their scope and a paired comparison with Holm correction. Appendix D reports participant-bootstrap intervals.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: §4.1 and Appendix A report that training ran on a single NVIDIA RTX PRO 6000 GPU for ≈144 hours (200 epochs × 2,000 steps, batch size 16). Latency benchmarks in §4.3 were measured on an NVIDIA L40S GPU with methodology details (200 segments, 10 warmup, `torch.cuda.Event` timing).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms with the NeurIPS Code of Ethics. All training and evaluation corpora are public speech datasets used under their licenses, and the listening study is described in Appendix D and in checklist items 14 and 15.

   Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Appendix G discusses reduced storage and transmission costs, potential downstream misuse for synthetic speech or impersonation, and privacy and domain-generalisation concerns.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The contribution is a compression model rather than a speech generator or a newly scraped dataset. Potential downstream misuse is discussed in Appendix G; no new high-risk generative model is introduced.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All datasets are cited with their original papers: CommonVoice (CC0), LibriTTS (CC-BY-4.0), VCTK (CC-BY-4.0), LibriSpeech (CC-BY-4.0). Baseline models (EnCodec, DAC, SpeechTokenizer, HiFi-Codec, Vocos, WavTokenizer, Mimi) are cited and used only for evaluation via their publicly released checkpoints under their respective open-source licences.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: GS-Codec code is released as part of the supplementary material, and audio samples are publicly available at `https://ronaluf.github.io/gs-codec/`. Training scripts and evaluation pipeline are included. Architecture, hyperparameters, and evaluation commands are fully documented in Appendix A.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [Yes]

Justification: Appendix D gives the listening protocol and the rating instructions. Participants were unpaid adult volunteers who took part anonymously; only anonymous ratings were recorded.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [No]

Justification: No institutional review was sought. The study consisted of anonymous quality ratings of short, publicly available speech recordings by adult volunteers; it collected no personal or sensitive information, involved no deception or intervention, and posed no foreseeable risk beyond ordinary listening at a self-chosen volume. Participants could stop at any time, and only completed sessions were analysed.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs were used only for writing and editing assistance and did not impact the core methodology, scientific rigor, or originality of the research. No LLMs are used as components of GS-Codec.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.