Towards Codec-LM Co-design for Neural Codec Language Models

Shih-Lun Wu[*] [1, 2] **Aakash Lahoti** [1, 3] **Arjun Desai** [1] **Karan Goel** [1]
Chris Donahue[†] [3] **Albert Gu**[†] [1, 3]

[1] Cartesia AI [2] MIT [3] CMU

[*] Work done during internship at Cartesia AI [†] Co-senior author **Correspondence:** slseanwu@mit.edu

Abstract

Neural codec language models (or *codec LMs*) are emerging as a powerful framework for audio generation tasks like text-to-speech (TTS). These models leverage advancements in language modeling and residual vector quantization (RVQ)-based audio codecs, which compress audios into discrete codes for LMs to process. Despite the close interdependence of codecs and LMs in these systems, research on codecs and LMs has largely remained siloed. In this work, we propose three techniques for better codec-LM co-design: (i) a *frame-wise codec encoder* that improves both LM log-likelihood and end-to-end TTS metrics, (ii) *LM codebook level dropout*, a method to efficiently navigate a portion of the codec-LM design space by training a single LM, and (iii) *increased codec frame duration*, which we show can accelerate inference while maintaining end-to-end performance. Our experiments demonstrate that combining all three co-design techniques results in doubled inference speed, and improvements in intelligibility, audio quality, and speaker control in TTS relative to a siloed baseline.

1 Introduction

Neural codec language models (or codec LMs) (van den Oord et al., 2017; Wu et al., 2024) have recently emerged as a prominent framework for text-to-speech (TTS) (Tan et al., 2021; Wang et al., 2023; Yang et al., 2024) and general audio generation tasks (van den Oord et al., 2016; Copet et al., 2023; Borsos et al., 2023; Yang et al., 2024), replacing autoregressive methods that model continuous raw waveforms (van den Oord et al., 2016; Kalchbrenner et al., 2018; Goel et al., 2022). The success of codec LMs can be attributed to improvements in the architecture, scaling, and efficiency of language models (LMs) (Vaswani et al., 2017; Brown et al., 2020; Dao et al., 2022; Gu and Dao, 2023), as well as increasingly high-fidelity convolutional audio codecs that employ the residual vector quantization (RVQ) technique (Zeghidour et al., 2021; Défossez et al., 2023; Kumar et al., 2023), bridging continuous-domain audio generation tasks with LM methods that model discrete tokens.

Although the codec and the LM are closely coupled, they represent relatively isolated research areas. Research on codecs (Zeghidour et al., 2021; Défossez et al., 2023; Kumar et al., 2023; Ahn et al., 2024) primarily focuses on achieving higher compression rates (i.e., lower bandwidths) while maintaining reconstruction quality. Conversely, research on codec-based LMs (Borsos et al., 2023; Wang et al., 2023; Copet et al., 2023; Yang et al., 2024) typically treats the codec as a fixed module and explores how to best model the codec tokens. (We defer more detailed Related Work to Appendix A.) While the design space of codecs and LMs combined is too large to explore exhaustively, considering each in isolation may be suboptimal when the goal is to improve the end-to-end performance.

In this work, we aim to break the isolation and uncover co-design principles between the codec and the LM. We identify several aspects that play a key role in the interactions between the two, and substantially impact the end-to-end generation quality and/or efficiency. Leveraging these co-design insights, we propose actionable interventions which can improve the performance and efficiency (both at training and inference) of end-to-end audio generation systems. Our technical contributions are:

- Considering the different impacts of receptive field overlaps in the RVQ codec encoder and decoder, we introduce a **framewise codec encoder** (Sec. 3.1), which encodes each frame (i.e., non-overlapping chunks in input audio) independently. We find that this leads to improvements in the LM log-likelihood (>8% higher), and all end-to-end TTS metrics (Table 1).
- Observing that the end-to-end generation per-

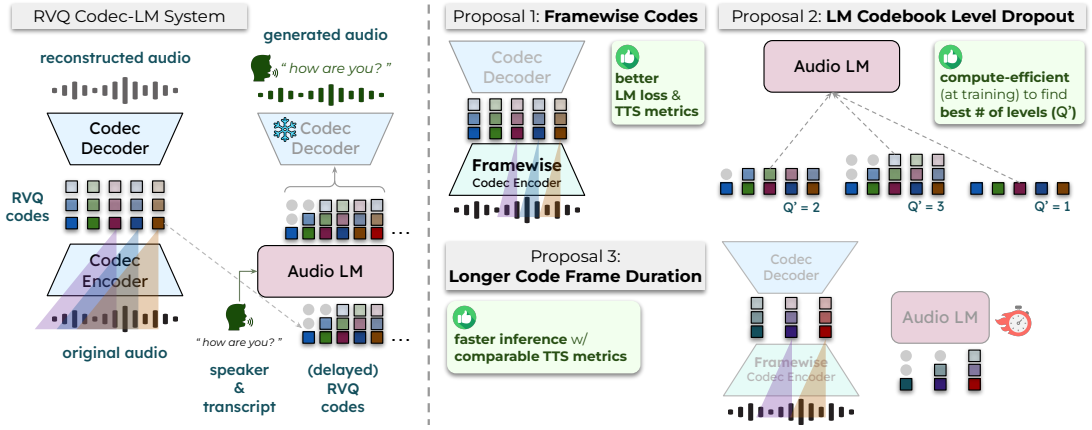

Figure 1: Overview of an RVQ-based codec-LM system for TTS (left), our contributions (right, **Proposals 1, 2 & 3**), and associated benefits. (Shaded triangles are receptive fields per code frame.)

formance is heavily influenced by number of RVQ codebook levels modeled by the LM, we propose **LM codebook level dropout** (Sec. 3.2), which allows practitioners to efficiently tune this salient hyperparameter of the codec-LM design space in a single LM training run (Fig. 2).

- As codec frame duration is inversely proportional to LM sequence length, we show that using **longer frame durations** (Sec. 3.3), while tuning other codec hyperparameters accordingly, can accelerate end-to-end TTS inference, and preserve TTS metrics (Table 2).

A schematic diagram of our end-to-end audio generation system and proposed techniques can be found in Fig. 1. Our experiments are based on a streamable (i.e., causal) variant of the DAC codec (Kumar et al., 2023), and we implement our changes (i.e., framewise encoder, and longer frame duration) without altering its architecture. We then train *Delay*-pattern LMs (Copet et al., 2023) for TTS, where LM codebook level dropout is applied, on the RVQ codes from our codecs. We finally demonstrate that combining all three co-design techniques doubles the end-to-end TTS inference speeds while *improving* all end-to-end TTS metrics (Table 3) concerning intelligibility, audio quality, and speaker control.

We open source our implementation of the framewise and causal DAC (Kumar et al., 2023) codecs at https://github.com/slSeanWU/descript-audio-codec/tree/main.

2 Technical Background

Residual vector quantization (RVQ)-based audio codecs. An RVQ-based audio codec compresses a continuous *waveform* $\boldsymbol{w} \in \mathbb{R}^{\mathrm{T f_s}}$, where T is the duration (in seconds) and $\mathrm{f_s}$ is the sampling rate (in Hz) of the waveform, into discrete *codes* $\boldsymbol{x} \in \mathcal{V}^{\mathrm{T f_x} \times Q}$. Here, $\mathcal{V} := \{1, 2, \ldots, |\mathcal{V}|\}$ represents the *codebook*, $\mathrm{f_x}$ (typically much smaller than $\mathrm{f_s}$) is the *frame rate* (in Hz) of the codec, and Q is the number of *codebook levels* used to represent each frame. We also call downsampling rate of the codec, i.e., $\mathrm{f_s}/\mathrm{f_x}$, the *frame size* (an integer number of audio *samples*) and $1/\mathrm{f_x}$ the *frame duration* (in seconds). The term *residual* refers to how the Q codebook levels are structured to progressively refine the quantization (Zeghidour et al., 2021). Let the unquantized representation (i.e., the codec encoder output) for the i-th frame be denoted by $\boldsymbol{h}_i^{(1)} \in \mathbb{R}^D$, where D is the codec encoder's output dimension. The RVQ process works iteratively for each level $q \in \{1, \ldots, Q\}$ on a frame-by-frame basis, quantizing the residual information from preceding levels using a level-wise learned codebook $\mathcal{C}^{(q)} : \mathcal{V} \to \mathbb{R}^D$. The operations at each level are:

$$x_{i,q} := \arg\min_{\tilde{x} \in \mathcal{V}} \|\boldsymbol{h}_i^{(q)} - \mathcal{C}^{(q)}(\tilde{x})\|_2^2 \quad (1)$$

$$\boldsymbol{h}_i^{(q+1)} := \boldsymbol{h}_i^{(q)} - \mathcal{C}^{(q)}(x_{i,q})\,, \quad (2)$$

where $x_{i,q} \in \mathcal{V}$ is an element in the code sequence $\boldsymbol{x}$, and $\mathcal{C}^{(q)}(x_{i,q}) \in \mathbb{R}^D$ is the quantized representation corresponding to $x_{i,q}$. The level-wise quantized representations are summed frame-by-frame, i.e., $\sum_q^Q \mathcal{C}^{(q)}(x_{i,q})$; $\forall i \in \{1, \ldots, \mathrm{T f_x}\}$, and sent to the decoder to reconstruct the original waveform.

Typically, during RVQ codec training, *quantizer dropout* (Zeghidour et al., 2021; Kumar et al., 2023) is applied, which sometimes performs Eqn. (1) and (2) for $Q_{\text{trunc}} < Q$ levels. This enables the codec to encode and reconstruct audio waveforms at all Q possible RVQ level counts.

Language modeling with *Delay* pattern of RVQ codes. We can construct an end-to-end audio generative model by training an LM on the RVQ codes $\boldsymbol{x} \in \mathcal{V}^{\mathrm{Tf_x} \times Q'}$, where $Q' \in \{1, \ldots, Q\}$ is a subset of the RVQ levels to model. To model such 2D-structured codes, we adopt the *Delay* pattern proposed in (Copet et al., 2023), which makes a good tradeoff between the efficiency and efficacy of modeling the RVQ codes $\boldsymbol{x}$. Instead of naively flattening $\boldsymbol{x}$ to a sequence of $\mathrm{Tf_x} \times Q'$ elements, it shifts the q-th level of $\boldsymbol{x}$ to the right by q positions, creating a shifted code sequence $\boldsymbol{x}^{(\mathrm{delay})} \in \mathcal{V}^{(\mathrm{Tf_x}+Q'-1) \times Q'}$, where each frame is $\boldsymbol{x}_t^{(\mathrm{delay})} := [x_{t-q+1,\, q}]_{q=1}^{Q'}$. Then, the LM models:

$$p(\boldsymbol{x}) = p(\boldsymbol{x}^{(\mathrm{delay})}) := \prod_{t=1}^{\mathrm{Tf_x}+Q'-1} p(\boldsymbol{x}_t^{(\mathrm{delay})} \mid \boldsymbol{x}_{<t}^{(\mathrm{delay})}), \tag{3}$$

predicting the elements in each frame $\boldsymbol{x}_t^{(\mathrm{delay})}$ in parallel. Though omitted in Eqn. (3), the LM is typically trained with conditions $\boldsymbol{y}$ expected from the user, e.g., text transcripts and speaker characteristics. Bringing all components together, our codec-LM audio generation system models:

$$p(\boldsymbol{w}, \boldsymbol{x} \mid \boldsymbol{y}) := \underbrace{p(\boldsymbol{w} \mid \boldsymbol{x})}_{\text{learned by codec}} \cdot \underbrace{p(\boldsymbol{x} \mid \boldsymbol{y})}_{\text{learned by LM}}, \tag{4}$$

where conditional independence between waveform $\boldsymbol{w}$ and user inputs $\boldsymbol{y}$ is assumed given codes $\boldsymbol{x}$. We note that $p(\boldsymbol{w} \mid \boldsymbol{x})$ is typically a deterministic mapping parameterized by the RVQ codec decoder.

3 Method

3.1 Codes with non-overlapping receptive fields (*Framewise codec encoder*)

Most common RVQ audio codecs (Zeghidour et al., 2021; Défossez et al., 2023; Kumar et al., 2023) set the stride size of each 1D convolutional layer to be smaller than the filter size. This way the neighboring outputs (along the time dimension) have overlapping receptive fields. When we consider the entire codec encoder, where multiple convolutional layers are stacked, this overlapping property at each layer causes the receptive field of each code frame $\boldsymbol{x}_t$ to overlap with those of preceding code frames $[\boldsymbol{x}_{t-k}, \ldots, \boldsymbol{x}_{t-1}]$, assuming the codec is causal.[1] A similar property also holds in the codec decoder, i.e., each sample in the reconstructed waveform $\hat{\boldsymbol{w}}$ is influenced by multiple code frames.

[1]For example, for the architecture of DAC (Kumar et al., 2023), the extent of overlap is $k = 8$.

If we reason about the frame-level overlaps, it is intuitive that they benefit the decoder, as the mutual information between multiple code frames can be leveraged for improved reconstruction. On the other hand, whether these overlaps are advantageous on the encoder side is less clear. They may provide the opportunity for the codec to pack information in high-complexity waveform segments (e.g., fast speech with frequent intonation changes) into neighboring code frames corresponding to low-information segments (e.g., silence), hence improving audio reconstruction. However, this could be detrimental for the downstream LM as each code frame may hold varying amounts of (confounding) information from preceding frames.

Therefore, we propose a setup where the codes are encoded *framewise*, i.e., each code frame $\boldsymbol{x}_t$ has a receptive field covering only $\mathrm{f_s}/\mathrm{f_x}$ waveform samples, without overlapping with other code frames. Operationally, this is achieved by reshaping the waveform (i.e., the inital input to the codec encoder) from $(B, \mathrm{Tf_s}, 1)$, where the dimensions represent (batch, sequence, channels), to $(B\mathrm{Tf_x},\, \mathrm{f_s}/\mathrm{f_x},\, 1)$. Since the downsampling rate of the entire encoder is precisely $\mathrm{f_s}/\mathrm{f_x}$, the final encoder output is of shape $(B\mathrm{Tf_x}, 1, D)$, which we then reshape back to $(B, \mathrm{Tf_x}, D)$ before quantization as in normal codecs with frame-level overlaps. Note that no architectural changes are required.

This setup with *encoder-framewise* and *decoder-overlapping* receptive fields retains desirable properties such as leveraging mutual information between code frames for reconstruction, Also, the information unique to each frame of waveform samples is encoded *distinctly* into one code frame, instead of spilling over multiple code frames, which we anticipate might benefit the downstream LM.

3.2 LM Codebook level dropout (*CL drop*)

Here we propose a novel method designed to increase the efficiency of hyperparameter tuning for the number of codec RVQ levels Q' used when training the downstream LM. The choice of the hyperparameter Q' can have a substantial impact on the end-to-end audio generation performance of the codec LM system. While increasing Q' monotonically improves codec audio reconstruction due to a wider information bottleneck, its impact on the combined codec LM system is ambiguous. Using too low of a Q' value in the LM could result in poor audio quality, while using too high of a value could be detrimental as modeling finer-grained lev-

	Framewise Enc. ?	*Codec Recons.*	*Text-to-Speech*				*Uncond. Music*	
Codec setting		Mel-L1↓	NLL↓	WER↓	NISQA↑	Spk. sim.↑	NLL↓	FAD↓
Causal	✗	**.846**	5.46±.00	4.12±.35	4.35±.01	80.2±.1	6.06±.01	18.7±1.2
Proposed	✓	.873	**4.97**±.02	**3.71**±.19	**4.37**±.02	**80.7**±.2	**5.16**±.00	**17.1**±0.8

Table 1: Codec encoder receptive field settings vs. end-to-end TTS & music generation performance. Our proposed **framewise codec encoder** (Sec. 3.1) consistently beats the commonly used streamable setting (i.e., *Causal*) both on LM likelihood (cf. NLL) and all end-to-end metrics. Stdev over 5 runs follow ±.

els may: (i) present information that is too stochastic for the LM to process effectively, or (ii) shift the LM's capacity away from the coarser-grained levels which contain more crucial structural or semantic information about the audio.[2]

However, naively training $\mathcal{O}(Q)$ LMs to tune Q' is computationally expensive. Thus, we propose *codebook level dropout* (CL drop), which trains just a single LM that allows evaluation/inference at all possible level counts up to Q, analogous to the quantizer dropout method used to train the codec. To perform CL drop, we first define a *dropout distribution* $\mathcal{P}(q)$ over the all levels $q \in \{1, \ldots, Q\}$, and then during LM training, we truncate inputs $\boldsymbol{x}^{(\text{delay})}$ along the level dimension according to $\mathcal{P}(q)$. The LM's training objective can hence be written as:

$$\min_{\theta} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D},Q'\sim\mathcal{P}(q)} \left[-\log p_\theta \left(\boldsymbol{x}^{(\text{delay})}_{:,:Q'} \mid \boldsymbol{y} \right) \right] \tag{5}$$

where $\mathcal{D}$ is the LM training set with paired conditions $\boldsymbol{y}$ and RVQ codes $\boldsymbol{x}$ for the desired audio, and θ is the set of the LM's trainable parameters.

For CL drop to be effective in determining the best Q', its end-to-end performance profile across different level counts should closely mirror the trends without CL drop (i.e., the 'end-to-end TTS' curve in Fig. 3). Intuitively, the choice of $\mathcal{P}(q)$ is critical in preserving the trends, as it governs how much the LM's focus is shifted toward the lower (coarser-grained) levels.[3]

3.3 Navigating other codec hyperparameters

In addition to the number of RVQ levels (Q), there are two additional hyperparameters that affect the compression factor of the codec: (i) the codec's frame duration ($1/\text{f}_\text{x}$), and (ii) the codebook size ($|\mathcal{V}|$). The bitrate of the codec, equal to $Q\text{f}_\text{x}\log_2(|\mathcal{V}|)$ bits per second, is a function of these three factors and directly impacts the reconstruction quality. In siloed codec design, these three factors can be traded off freely to optimize for higher reconstruction quality at some fixed bitrate. However, in a co-design context, the LM's behavior can be impacted by different tradeoffs even when the codec's bitrate is kept fixed.

Here we make several observations about frame duration and codebook size respectively in the context of codec-LM co-design. From Eqn. (3), we can observe that the Delay LM sequence length is inversely proportional to frame duration. Thus, increasing it by a factor of two can roughly halve sequence length, resulting in efficiency gains and reduced inference latency. (Note that either $|\mathcal{V}|$ or Q should be increased accordingly to preserve audio quality.)

On the other hand, increasing the codebook size $|\mathcal{V}|$ may have mixed impacts on the LM. On the positive side, assuming the frame duration and bitrate are controlled, using a larger codebook (and hence fewer RVQ levels) reduces the extent of packing information from multiple (i.e., Q') code frames into one Delay LM timestep $\boldsymbol{x}^{(\text{delay})}_t$. However, increasing only $|\mathcal{V}|$ while holding Q constant leads to an exponential growth in the LM's vocabulary size (and embedding parameters) relative to a linear increase in bitrate. This growth can inflate the LM's memory footprint and introduce potential modeling challenges. Thus, while our CL drop technique can efficiently find the best Q' given a fixed $|\mathcal{V}|$, finding the optimal $|\mathcal{V}|$ still requires trial and error.

4 Experiments and Discussion

We first conduct experiments specifically for each proposed technique (i.e., Sec 3.1, 3.2, and 3.3) to elucidate their individual effects, and finally combine them to show their collective benefits. We use word error rate (WER), NISQA (Mittag et al., 2021), and cosine similarity of speaker embeddings (Jung et al., 2022) to evaluate the *intelligibility*, *audio quality*, and *speaker control* of end-to-end TTS generations. For music generation, we use Fréchet audio distance (FAD) (Kilgour et al., 2019) to capture overall quality. More experimental setups are deferred to Appendix C.

[2]Experiments on the impact Q' are in Appendix B.

[3]Experiments on different $\mathcal{P}(q)$'s are in Appendix D.

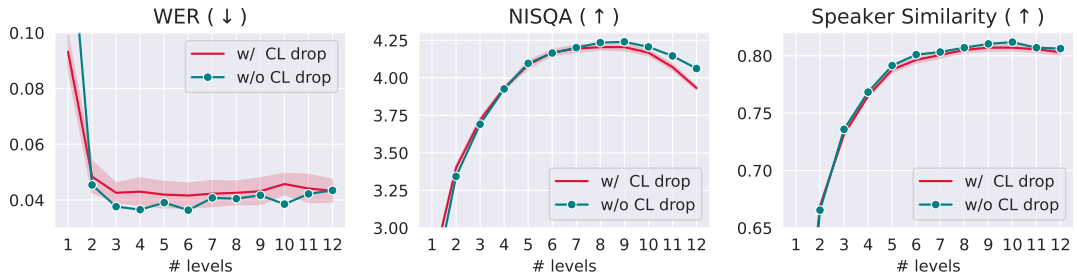

Figure 2: Number of RVQ codebook levels used by LM vs. end-to-end TTS metrics. Training one LM with **codebook level dropout** (‘**CL drop**’, Sec. 3.2) leads to performance trends that closely follow training $Q = 12$ LMs w/o CL drop at each $Q' \in \{1, \ldots, 12\}$. Note that practitioners can then train a second LM at the found optimal level count w/o CL drop for best possible performance. Shaded bands represent stdev over 3 runs.

Codec Config				*Codec Recons.*	*Text-to-Speech*			*Efficiency*
Frame dur.	$\log_2(\lvert\mathcal{V}\rvert)$	Q'	Rel. bitrate	Mel-L1↓	WER↓	NISQA↑	Spk. sim.↑	Inf. speedup↑
11ms	10	9	1.00×	.873	3.71 ±.19	4.37 ±.02	80.7 ±.2	1.00×
11ms	15	6	1.00×	.874	3.73 ±.28	4.33 ±.01	80.4 ±.1	1.01×
22ms	10	16	0.89×	.888	4.21 ±.33	**4.42** ±.01	**81.0** ±.1	1.94×
22ms	15	11	0.92×	.876	**3.55** ±.36	4.33 ±.01	79.3 ±.1	2.00×
44ms	10	32	0.89×	.875	6.73	4.14	76.7	3.20×
44ms	15	20	0.83×	.871	4.53	3.65	73.2	**3.77**×

Table 2: Effects of using **longer frame durations** (Sec. 3.3), holding audio reconstruction quality approximately constant by varying codebook size $|\mathcal{V}|$ and/or # of RVQ levels Q'. We measure the actual inference time (LM & codec decoding combined) over 50 samples with batch size 1 and treat the first row as the baseline for the ‘Inf. speedup’ column. In general, using a 2× frame duration (22ms) strikes best balance between performance and efficiency. Stdev over 5 runs follow ±. First row is the default configuration inherited from DAC.

Framewise codec encoder. Table 1 presents a comparison of audio reconstruction and downstream TTS (and music) generation performance with and without the use of our proposed framewise codec encoder. Here, we adopt the default DAC (Kumar et al., 2023) codec configurations.[4] Our framewise codec encoder setting **outperforms** the default streamable *causal* setting consistently, both **on LM likelihood** (>8% lower NLL) **and all end-to-end TTS and music generation metrics**. Notably, it is slightly worse on Mel-L1, underscoring the fact that **better audio reconstruction does not always translate to better end-to-end performance**. Due to its advantage, we conduct all subsequent experiments with framewise codec encoders, unless otherwise specified.

LM codebook level dropout (CL drop). Results of training the LM with codebook level dropout (see Sec. 3.2) are presented in Fig. 2. To examine how end-to-end performance evolves in the higher-bitrate regime, we use 15-bit codebooks ($\log_2(|\mathcal{V}|) = 15$) and codebook levels $Q = 12$ for the codec.[5] We experiment with various dropout distributions $\mathcal{P}(q)$ (see App. D for details) and conclude that it is best to train at the full level count (i.e., 12 in this case) for 90% of the steps and uniformly distribute the remaining 10% to all lower level counts. The curves in Fig. 2 show that training a single LM with CL drop produces a performance profile closely aligned with training 12 separate LMs without CL drop. This demonstrates that **CL drop is a reliable method for practitioners to efficiently optimize for the level count Q' with significantly reduced training compute**. Besides, the curves also show that WER, which focuses on (coarser) word-level information, reaches the best early at 3∼4 levels, while NISQA and speaker similarity, which are tied more closely to the fine-grained details, peak at around 9 levels. Though different metrics behave differently w.r.t. level count, we find that **choosing the best level count based on FAD** (shown in Fig. 3, which uses the same codec as here and would suggest using 9 levels) **achieves a balanced performance between all the TTS metrics** we consider.

[4] Frame duration $(1/\mathrm{f_x}) = 11$ms; number of RVQ levels (Q and Q') $= 9$; codebook size per level ($|\mathcal{V}|$) $= 2^{10}$.

[5] amounting to a max bitrate that is 2× that of official DAC.

Proposals			*Codec Config*			*Text-to-Speech Metrics*			*Efficiency*
#1	**#2**	**#3**	Frame dur.	$\log_2(\lvert\mathcal{V}\rvert)$	$Q' : Q$	WER↓	NISQA↑	Spk. sim.↑	Inf. speedup↑
✗	✗	✗	11ms	10	9 : 9	4.12 ±.35	4.35 ±.01	80.2 ±.1	1.00×
✓	✗	✗	11ms	10	9 : 9	**3.71** ±.19	4.37 ±.02	80.7 ±.2	1.01×
✓	✗	✓	22ms	10	16:16	4.21 ±.33	*4.42* ±.01	**81.0** ±.1	*1.95*×
✓	✓	✓	22ms	10	14:16	*3.86* ±.19	**4.43** ±.01	*80.8* ±.2	**2.01**×

Table 3: Combined improvements from using multiple proposed techniques—**#1**: Framewise codec encoder; **#2**: CL drop; **#3**: Longer frame duration. Q' denotes the # of levels the LM is trained with for end-to-end TTS, while Q denotes the RVQ codec's full # of levels. We *italicize* the second best setting for each metric. Compared to the baseline using a causal codec (1st row), applying all of our proposed techniques (last row) improves both the efficiency and all end-to-end TTS metrics.

Longer frame duration. Table 2 displays the effects of using longer frame durations ($\{1\times, 2\times, 4\times\}$ that of default DAC), and wider codebooks (2^{10} (default) or 2^{15} codewords per level). Here, we use the number of levels Q' (in this set of experiments, $Q' = Q$) as a variable to roughly control for audio reconstruction quality (i.e., Mel-L1). In general, **using a 22ms frame duration** (i.e., $2\times$ that of default DAC) **preserves or improves TTS performance and enjoys a $2\times$ inference speedup** at the same time. Increasing the frame duration to 44ms leads to substantially worse TTS metrics despite further efficiency gains. However, whether to increase the codebook size $|\mathcal{V}|$ from the default 2^{10} to accommodate longer frame durations remains unclear (better on WER, worse on other metrics), warranting a more fine-grained exploration (e.g., a dense sweep over 10- to 15-bit codebooks) in future work.

Combining all techniques. Table 3 illustrates the cumulative impact of progressively integrating our proposed techniques. In the last row, we apply LM codebook level dropout to a (22ms, 10-bit, 16-level) codec, identifying the optimal level count $Q' = 14$ using FAD on end-to-end TTS. Comparing the streamable baseline (1st row) and the final model with all our techniques (last row), we achieve substantial improvements across all end-to-end TTS metrics, while doubling inference speed.

Future work. Our work may be extended to: (i) study the theory of why framewise compressed representations improve language modeling, (ii) develop RVQ codecs that have flexibility also in codebook size and frame duration such that our LM codebook level dropout can be applied to multiple key hyperparameters altogether, and (iii) uncover the scaling properties (Hoffmann et al., 2022) of the optimal codec settings w.r.t. larger models and more training data.

References

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

Sunghwan Ahn, Beom Jun Woo, Min Hyun Han, Chanyeong Moon, and Nam Soo Kim. 2024. HILCodec: High fidelity and lightweight neural audio codec. *arXiv preprint arXiv:2405.04752*.

Dmitry Bogdanov, Minz Won, Philip Tovstogan, Alastair Porter, and Xavier Serra. 2019. The MTG-Jamendo dataset for automatic music tagging. In *Proc. Workshop on Machine Learning for Music Discovery (ML4MD)*.

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

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems (NeurIPS)*.

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

Jade Copet, Felix Kreuk, Itai Gat, Tal Remez, David Kant, Gabriel Synnaeve, Yossi Adi, and Alexandre Défossez. 2023. Simple and controllable music generation. *Advances in Neural Information Processing Systems (NeurIPS)*.

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

Tri Dao and Albert Gu. 2024. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *Proc. Int. Conf. on Machine Learning (ICML)*.

Michaël Defferrard, Kirell Benzi, Pierre Vandergheynst, and Xavier Bresson. 2017. FMA: A dataset for music analysis. In *Proc. Int. Soc. for Music Information Retrieval Conf. (ISMIR)*.

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

Alexandre Défossez, Laurent Mazaré, Manu Orsini, Amélie Royer, Patrick Pérez, Hervé Jégou, Edouard Grave, and Neil Zeghidour. 2024. Moshi: a speech-text foundation model for real-time dialogue. Technical report, Kyutai.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. NAACL*.

Chris Donahue, Julian McAuley, and Miller Puckette. 2019. Adversarial audio synthesis. In *Proc. Int. Conf. on Learning Representations (ICLR)*.

Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. 2017. Audio set: An ontology and human-labeled dataset for audio events. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*.

Karan Goel, Albert Gu, Chris Donahue, and Christopher Ré. 2022. It's raw! audio generation with state-space models. In *Proc. Int. Conf. on Machine Learning (ICML)*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *Advances in Neural Information Processing Systems (NeurIPS)*.

Albert Gu and Tri Dao. 2023. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*.

Ali Hatamizadeh and Jan Kautz. 2024. Mambavision: A hybrid mamba-transformer vision backbone. *arXiv preprint arXiv:2407.08083*.

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

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. 2021. Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM Trans. on Audio, Speech, and Language Processing (T-ASLP)*.

Shengpeng Ji, Minghui Fang, Ziyue Jiang, Rongjie Huang, Jialung Zuo, Shulei Wang, and Zhou Zhao. 2024. Language-Codec: Reducing the gaps between discrete codec representation and speech language models. *arXiv preprint arXiv:2402.12208*.

Jee-weon Jung, You Jin Kim, Hee-Soo Heo, Bong-Jin Lee, Youngki Kwon, and Joon Son Chung. 2022. Pushing the limits of raw waveform speaker recognition. In *Proc. Interspeech*.

Nal Kalchbrenner, Erich Elsen, Karen Simonyan, Seb Noury, Norman Casagrande, Edward Lockhart, Florian Stimberg, Aaron Oord, Sander Dieleman, and Koray Kavukcuoglu. 2018. Efficient neural audio synthesis. In *Proc. Int. Conf. on Machine Learning (ICML)*.

Kevin Kilgour, Mauricio Zuluaga, Dominik Roblek, and Matthew Sharifi. 2019. Fréchet audio distance: A reference-free metric for evaluating music enhancement algorithms. In *Proc. Interspeech*.

Jaehyeon Kim, Keon Lee, Seungjun Chung, and Jaewoong Cho. 2024. CLam-TTS: Improving neural codec language model for zero-shot text-to-speech. In *Proc. Int. Conf. on Learning Representations (ICLR)*.

Yuma Koizumi, Heiga Zen, Shigeki Karita, Yifan Ding, Kohei Yatabe, Nobuyuki Morioka, Michiel Bacchiani, Yu Zhang, Wei Han, and Ankur Bapna. 2023. LibriTTS-R: A restored multi-speaker text-to-speech corpus. In *Proc. Interspeech*.

Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. 2020. HiFi-GAN: Generative adversarial networks for efficient and high fidelity speech synthesis. *Advances in Neural Information Processing Systems (NeurIPS)*.

Kundan Kumar, Rithesh Kumar, Thibault De Boissiere, Lucas Gestin, Wei Zhen Teoh, Jose Sotelo, Alexandre De Brebisson, Yoshua Bengio, and Aaron C Courville. 2019. MelGAN: Generative adversarial networks for conditional waveform synthesis. *Advances in Neural Information Processing Systems (NeurIPS)*.

Rithesh Kumar, Prem Seetharaman, Alejandro Luebs, Ishaan Kumar, and Kundan Kumar. 2023. High-fidelity audio compression with improved RVQGAN. *Advances in Neural Information Processing Systems (NeurIPS)*.

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *Proc. Int. Conf. on Learning Representations (ICLR)*.

Gabriel Mittag, Babak Naderi, Assmaa Chehadi, and Sebastian Möller. 2021. NISQA: A deep cnn-self-attention model for multidimensional speech quality prediction with crowdsourced datasets. In *Proc. Interspeech*.

Gautham J Mysore. 2014. Can we automatically transform speech recorded on common consumer devices in real-world environments into professional production quality speech?—a dataset, insights, and challenges. *IEEE Signal Processing Letters*.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2023. Robust speech recognition via large-scale weak supervision. In *Proc. Int. Conf. on Machine Learning (ICML)*.

Zafar Rafii, Antoine Liutkus, Fabian-Robert Stöter, Stylianos Ioannis Mimilakis, and Rachel Bittner. 2017. The MUSDB18 corpus for music separation.

Xu Tan, Tao Qin, Frank Soong, and Tie-Yan Liu. 2021. A survey on neural speech synthesis. *arXiv preprint arXiv:2106.15561*.

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

Aaron van den Oord, Oriol Vinyals, et al. 2017. Neural discrete representation learning. *Advances in Neural Information Processing Systems (NeurIPS)*.

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

Roger Waleffe, Wonmin Byeon, Duncan Riach, Brandon Norick, Vijay Korthikanti, Tri Dao, Albert Gu, Ali Hatamizadeh, Sudhakar Singh, Deepak Narayanan, et al. 2024. An empirical study of mamba-based language models. *arXiv preprint arXiv:2406.07887*.

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

Haibin Wu, Xuanjun Chen, Yi-Cheng Lin, Kai-wei Chang, Ho-Lam Chung, Alexander H Liu, and Hung-yi Lee. 2024. Towards audio language modeling–an overview. *arXiv preprint arXiv:2402.13236*.

Dongchao Yang, Jinchuan Tian, Xu Tan, Rongjie Huang, Songxiang Liu, Haohan Guo, Xuankai Chang, Jiatong Shi, Sheng Zhao, Jiang Bian, Zhou Zhao, Xixin Wu, and Helen M. Meng. 2024. UniAudio: Towards universal audio generation with large language models. In *Proc. Int. Conf. on Machine Learning (ICML)*.

Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. 2021. Soundstream: An end-to-end neural audio codec. *IEEE/ACM Trans. on Audio, Speech, and Language Processing (T-ASLP)*.

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

A Related Work

Neural audio codecs. Compressing and quantizing long, continuous audio waveforms into shorter discrete codes using a convolutional autoencoder was first proposed by van den Oord et al. (2017). Their proposed VQ-VAE method involves online K-Means for quantizing latent representations and a reconstruction objective on the decoder's output. Later, SoundStream (Zeghidour et al., 2021) introduced the 2D-structured Residual Vector Quantization (RVQ) to such codecs. This work also integrated a mixture of discriminators, a technique adoped from GAN-based audio synthesis (Goodfellow et al., 2014; Donahue et al., 2019; Kumar et al., 2019; Kong et al., 2020), on top of the decoder to enhance the perceptual quality of reconstructed waveforms—this RVQ-GAN setup has since been a norm for neural audio codecs. EnCodec (Défossez et al., 2023) and DAC (Kumar et al., 2023) further advanced the RVQ-GAN architecture with optimized discriminator setup, activation function, and (low) latent dimensionality. HILCodec (Ahn et al., 2024) showed that layer-wise variance constraining helps with the depth scaling of lightweight RVQ-GAN codecs. Overall, research in neural audio codecs has focused on achieving higher compression (i.e., lower bitrates) while maintaining audio reconstruction quality, rather than downstream audio generation, and often involved detailed architectural designs and tuning. In contrast, our work approaches codec design from an end-to-end audio generation practitioners' perspective, exploring codec hyperparameters that are both easily configurable and influential to the end-to-end system.

LM-based end-to-end audio generation. Autoregressive modeling of compressed discrete codes for audio waveforms was first proposed alongside VQ-VAE (van den Oord et al., 2017). AudioLM (Borsos et al., 2023) introduced a hierarchical LM approach that first generates *semantic tokens* (Hsu et al., 2021; Chung et al., 2021), derived from BERT-like pretraining (Devlin et al., 2019) on audio data, followed by RVQ codes (or *acoustic tokens*), resulting in better long-term coherence in generated audios. To navigate the efficiency-quality tradeoff given an RVQ codec, VALL-E (Wang et al., 2023) proposed non-autoregressive modeling for all RVQ levels except the coarsest one, and MusicGen (Copet et al., 2023) introduced the *Delay* pattern, dramatically shortening the sequence length while preserving key autoregressive dependencies. UniAudio (Yang et al., 2024) unified tokenization schemes for text, phonemes, audio, and symbolic music to build an LM for a wide range of audio generation tasks. Despite these advancements, all aforementioned work treated the audio codec, which is upstream from the LM, as a fixed component, leaving out the potential gains from a co-design between the codec and the LM.

Co-design of audio codecs and LMs. Compared to the two previously discussed areas, designing codecs with the goal of improving end-to-end audio generations is a relatively nascent direction. SpeechTokenizer (Zhang et al., 2024) proposed to distill information in semantic tokens (Hsu et al., 2021) into the first (coarsest) level of the RVQ codec, alleviating the need of using two LMs (Borsos et al., 2023; Agostinelli et al., 2023) in tandem for semantic and acoustic RVQ tokens. Moshi (Défossez et al., 2024), a work conducted concurrently with ours, adopted this technique and used a causal codec setup to enable low-latency, streamable real-time voice conversations. Language-Codec (Ji et al., 2024) proposed to arrange the RVQ levels in a first-parallel, then-sequential fashion to distribute information more evenly among the RVQ levels. While the methods above improved the latency and/or quality of end-to-end generations, they focused on single, and highly specific, modifications to the codec. Meanwhile, our work investigate the downstream impact of multiple general RVQ codec hyperparameters in combination, painting a more complete picture for end-to-end system practitioners.

B Impact of RVQ levels on reconstruction vs. on end-to-end generation

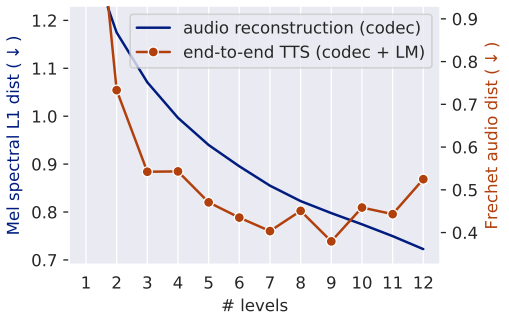

Figure 3: Impacts of # of codebook levels Q' are different on codec-only *audio reconstruction* vs. *end-to-end TTS* involving both the codec and the LM. (frame duration $1/\mathrm{f_x} = 11$ms; codebook size $|\mathcal{V}| = 2^{15}$.)

We train a single RVQ codec on speech data

with $Q = 12$ levels and train 12 LMs for text-to-speech (TTS) using each possible value of $Q' \in \{1, \ldots, 12\}$. In Fig. 3, we first plot the codec audio reconstruction performance as measured by Mel-spectral L1 distance. We also plot the end-to-end codec LM system performance as measured by Fréchet audio distance (FAD) (Kilgour et al., 2019), an end-to-end metric for audio generation. We observe that end-to-end performance improves as the number of levels increases towards a global minima at 9 levels and deteriorates afterwards, as opposed to the monotonically improving curve of audio reconstruction.

C Experimental Setup

Datasets for codec. For TTS, we collect 1.7K hours of YouTube podcast data in-house to train the codec. For music experiments, we use the *medium* version of FMA dataset (Defferrard et al., 2017) containing 200 hours of multitrack music. To evaluate audio reconstruction of our codecs, we follow DAC (Kumar et al., 2023) and create a dataset of 3K 10-second audios comprising speech (Mysore, 2014), music (Rafii et al., 2017) and general sounds (Gemmeke et al., 2017) (1K each).

Datasets for LM. For TTS, we use the 550-hour LibriTTS-R (Koizumi et al., 2023) for LM training, and its *test-clean* split (8 hours, 4.7K samples) for evaluation. For unconditional music generation, we train our LMs on 1.5K hours of multitrack music from MTG-Jamendo dataset (Bogdanov et al., 2019). We exclude examples with vocals using the associated metadata, and and hold out 1.5K examples for evaluation.

Codec model specifics. We utilize the open-source code of DAC (Kumar et al., 2023) and implement our changes on top. Our codecs have 76∼84M non-codebook parameters due to various frame durations. We train our codecs for 300K steps with an effective batch size of 75 seconds of audio. We use the AdamW (Loshchilov and Hutter, 2018) optimizer with 10^{-4} initial learning rate and exponential decay. The training process takes about 25 hours on 4 NVidia H100 (80G) GPUs.

LM model specifics. Following recent validation that a hybrid of state-space model (SSM) and attention outperforms either approach alone (Waleffe et al., 2024; Hatamizadeh and Kautz, 2024), we use 24 layers of stacked Mamba2 (Dao and Gu, 2024) and Transformer decoder blocks (Vaswani et al., 2017), totaling 400M non-embedding parameters. We prepend the conditioning information for TTS (i.e., $\boldsymbol{y}$, which includes text transcripts and speaker embedding) to the RVQ audio codes $\boldsymbol{x}^{(\text{delay})}$. The text transcript is transformed into character embeddings, while the speaker embedding is extracted using a raw waveform-based speaker recognition model (Jung et al., 2022).

We train our LMs for 30K steps with a batch size equivalent to 500 seconds of audio. We use the AdamW optimizer (Loshchilov and Hutter, 2018) with a peak learning rate of 4×10^{-4}, and 10% warmup steps followed by cosine decay. Training takes 12 hours on 8 H100 (80G) GPUs. For inference, we use pure sampling from the LM's output logits.

Evaluation for audio reconstruction (codec). We follow (Kumar et al., 2023) and compute the L1 distance between the log-scaled Mel spectrograms of the original and reconstructed waveforms to measure reconstruction at the signal level. We abbreviate this metric as *Mel-L1* hereafter.

Evaluation for end-to-end audio generation (codec + LM). To evaluate our end-to-end TTS system involving both the codec and the LM, we consider the following three aspects:

- **Intelligibility:** Following (Wang et al., 2023), we measure the word error rate (WER, in %) between the given text transcript and automatically transcribed text by Whisper (Radford et al., 2023) (v3 large) model from the generated speech.
- **Audio quality:** We leverage NISQA (Mittag et al., 2021) overall quality score, which is predicted by a CNN-Transformer model trained on pairs of speech audios and human-labeled quality scores in the range of $[1, 5]$. NISQA has been shown to correlate well (Pearson's $r \geq 0.9$) with human judgments of speech audio quality.
- **Speaker control:** Following (Wang et al., 2023; Kim et al., 2024), we compute the cosine similarity ($\in [-1, 1]$, reported in %) between the given speaker embedding and that extracted from the generated speech, using the same speaker recognition model (Jung et al., 2022).

For experiments on unconditional music generation, following (Copet et al., 2023; Agostinelli et al., 2023), we report Fréchet audio distance

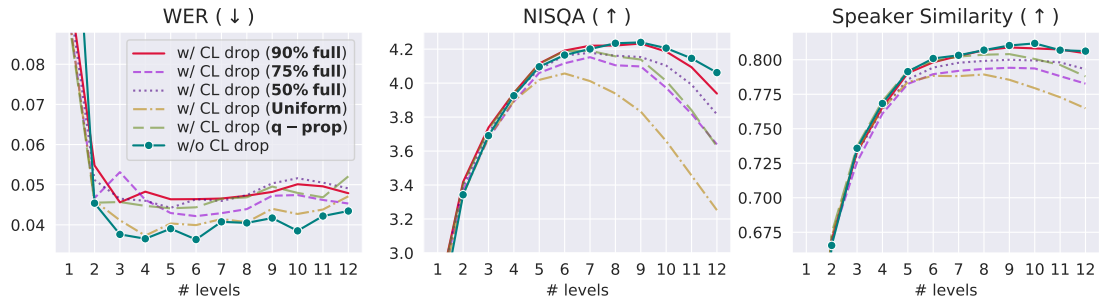

Figure 4: Effects of using different dropout distributions, i.e., $\mathcal{P}(q)$, for LM codebook level dropout. The curves of 'w/ CL drop' settings are the closer to those of 'w/o CL drop' the better.

(FAD) (Kilgour et al., 2019) computed on audio embeddings from the VGGish (Hershey et al., 2017) audio classification model. FAD captures how realistic the generations are at the dataset level (i.e., all generations vs. all reference inputs) using feature-wise covariances estimated from all audio embeddings of the generated/reference set.

D Choosing A Good $\mathcal{P}(q)$ for LM Codebook Level Dropout

For LM codebook level dropout (i.e., CL drop) to be effective in determining the optimal level count, its performance profile w.r.t. the level count should trend as closely as possible to that resulting from training LMs without CL drop at every possible number of levels. Here, we find that the choice of dropout distribution $\mathcal{P}(q)$, which determines the fraction of training steps allocated to each level count, to be critical. We experiment with a total of 5 different $\mathcal{P}(q)$'s detailed below:

- **Uniform:** $\mathcal{P}(q) := \frac{1}{Q};\ \forall q \in \{1, \ldots, Q\}$, i.e., every level count gets equal attention.
- **q-proportional (or *q-prop*):** $\mathcal{P}(q) := \frac{q}{Z(Q)};\ \forall q \in \{1, \ldots, Q\}$, where the normalization constant $Z(Q) := \sum_{q'=1}^{Q} q'$, i.e., the fraction for each level count q is proportional to q.
- **50% full:** $\mathcal{P}(q) := 0.5$ for $q = Q$, and $\mathcal{P}(q) := \frac{1-0.5}{Q-1};\ \forall q \in \{1, \ldots, Q-1\}$, i.e., the full level count Q gets 50% of the steps, and all the lower level counts share the remaining 50% uniformly.
- **75% full:** $\mathcal{P}(q) := 0.75$ for $q = Q$, and $\mathcal{P}(q) := \frac{1-0.75}{Q-1};\ \forall q \in \{1, \ldots, Q-1\}$, which is similar to **50% full** but focuses more on the full level count Q.
- **90% full:** $\mathcal{P}(q) := 0.9$ for $q = Q$, and $\mathcal{P}(q) := \frac{1-0.9}{Q-1};\ \forall q \in \{1, \ldots, Q-1\}$, which puts even more focus on $q = Q$ than **75% full**.

The performance profiles resulting from these $\mathcal{P}(q)$'s are shown in Fig. 4. The NISQA (which evalutes *audio quality*) and speaker similarity profiles suggest that **90% full** is the best choice among the five $\mathcal{P}(q)$'s. Other choices all peak at relatively lower level counts, and **Uniform**, which is the most straightforward option, appears to be the worst of the five.

The reasons behind why allocating only 10% to lower level counts leads to metrics that track most closely those from training separate LMs for each level count are left for further investigation. Our intuition is that, training with Q levels already includes modeling all the lower levels, and hence the LM only needs a small number of steps to adapt to the scenarios where the finer-grained information in higher levels is absent.