LOW RESOURCE AUDIO CODEC CHALLENGE Sublime SYSTEM DESCRIPTION

Piotr Dura
piotrdura7@gmail.com

***Abstract*—This work aims to advance neural audio coding by exploring novel approaches for Speech Vocoding and Vector Quantization (VQ). Both Track 1 and Track 2 systems are proposed, and both are convolutional encoder-decoder models with discrete representation emitted by the encoder. The decoder is a conv1d-conv2d hybrid Fourier-domain vocoder we call Sublime. Both Tracks share the same Vocoder weights. A novel quantization scheme, which we call Simulated Annealing Vector Quantization (SAVQ), is proposed along with a method to prevent codebook collapse.**

***Index Terms*—LRAC 2025, audio coding, VQ, generative adversarial networks**

I. INTRODUCTION

In this work, we present the design of a participant system for the 2025 LRAC challenge Tracks 1 and 2. Track 1 system is comprised of the encoder (3.8M params, 399.7 MFLOPS) and decoder (2.5M params, 294.1 MFLOPS). Track 2 system also contains a frontend (20.6M params, 2284.6 MFLOPS). Quantizer can operate in two modes — 1kbps and 6kbps, both modes can be used interchangeably by the decoder. The model is fully causal, but the buffering latency of analysis-synthesis accounts for the full 30ms end-to-end latency budget. Track 2 reuses the decoder weights, and instead of the encoder-SAVQ combination, a separate convolutional encoder is trained with the objective of predicting the codes via a Cross Entropy Loss. Track 2 encoder has an additional 20ms of algorithmic latency which result in a 50ms end-to-end latency. The latency figures are not estimated, but are the worst-case, measured latencies imposed by the algorithm. Presented MFLOPS numbers are obtained using a pytorch calflops package.

Total amount of training time spent on both Tracks is less than 120 gpu-hours on an NVIDIA RTX 4090, out of which 96 gpu-hours were assigned for Track 1 and 24 gpu-hours for Track 2.

II. ENCODER

The first processing stage converts the input 24kHz mono waveform into two log-mel spectrograms (10ms hop, 20ms window, 64 filters and 10ms hop, 30ms window, 96 filters) and concatenates them in channel dimension. The result is processed by a conv1d block (kernel size 3, 160 input channels, 256 hidden channels, 120 feed-forward channels), then the frames are stacked with stride $N = 2$ to form a 20ms-per-frame sequence. Causal stacking is used to not increase the latency, so that the initial stacked frames are partial during inference. The stacked sequence is further processed by 8 conv1d blocks (each has kernel size 3, 384 hidden channels, 384 feed-forward channels) and projected into query vectors q. Conv1d blocks are inspired by ConvNeXt [1] but use vanilla (non-depth-separable) conv1d and include a transformer-style feed-forward block with expansion and RMSNorm normalization.

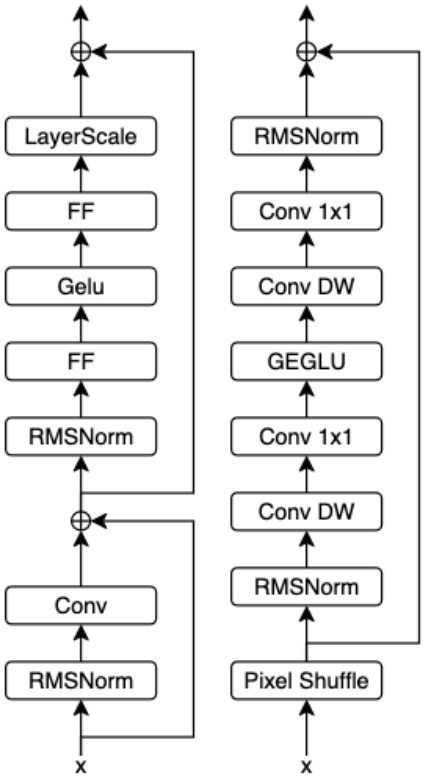

Fig. 1. Conv1d and Conv2d blocks.

III. QUANTIZATION

Standard quantization schemes require finding a nearest-neighbor embedding out of an embedding table for each frame of the input using an L2 or cosine distance. Since the embedding lookup is nondifferentiable, a straight-through estimation (STE) is typically used. An optional commitment loss can be used to penalize the distance between the input frames and the quantized output frames. Typical implementations leverage techniques like K-means initialization, dead code revival or smoothing of the update of an embedding table via an EMA. To improve the efficiency of compression,

SoundStream introduces a residual VQ [4]. RVQ quantizes the input and then iteratively quantizes the resulting quantization error with a number of separate embedding tables. More recent approaches such as FSQ [5] avoid using an embedding table altogether.

The proposed SAVQ utilizes cosine Cross-Attention with a learnable bank of embeddings and is parametrized by the temperature T:

$$\text{SAVQ}(q, k, v; T) = \text{softmax}\left(\frac{\sqrt{D}\ \text{cosine\_sim}(q, k)}{T}\right) V$$

where $k, q \in \mathbb{R}^D$

As the training progresses the temperature is annealed with a fixed annealing schedule. In the early stages when the temperature is high, attention over the embeddings has high entropy. Over time the sharpness of the attention increases and the behavior of the system shifts towards compression. As temperature approaches zero, attention over the embeddings approaches a one-hot vector. Notice, that a standard dot-production attention would not be effective for this purpose, as the network would be able to arbitrarily parametrize norms of query-key pairs. Second, because the cosine metric is used the normalization term of $\sqrt{D}$ is moved to the numerator. To increase the efficiency of compression, G groups of embeddings have been used, which is equivalent to a multi-head attention.

To enable efficient learning of the encoder even with low temperatures a temperature floor parameter T_F is introduced. Activations that are emitted by the quantizer are calculated using the original T, only the gradient that flows back to the encoder is modified as if the attention weights were calculated using $max(T, T_F)$.

This formulation, while empirically effective, suffered from codebook collapse, where roughly 10-20% of all codes ended up never being the top-1 activation. As the training progressed and temperature was annealed these codes were never reused by the model. A simple technique would be to employ entropy maximization loss:

$$L_H(p) = -H(p) = \sum_{t,k} p(t,k)\ log\ p(t,k)$$

where p is a categorical distribution over codes in a given codebook, t is batch-time-step, k is embedding index.

Since we don't want to penalize low codebook entropy as long as all codes have non-zero usage, we applied an ad-hoc loss called reciprocally-weighted smoothed surprisal (RWSS):

$$RWSS(p) = -\sum_{k} \frac{1}{p_K(k) + \epsilon} \sum_{t \in Q_q(p,k)} log\ p(t,k)$$

where $p_K(k)$ denotes empirical probability that the code k is a top-1 activation calculated over batch examples and time steps, ϵ is a smoothing constant, Q_q is a set of batch-time-steps that contains upper q-quantile of all $p(:, k)$

Intuitively, entropy maximization would penalize high log-probabilities of "activated" tokens and would move the attention weights towards a uniform distribution. RWSS loss penalizes low log-probabilities of tokens that are rarely activated (low $p_K(k)$) and routes that penalty only to the frames that already have high contribution of those codes. Version of this loss that penalized all frames instead of the top q-quantile resulted in a codebook in which code utilization oscillated highly over time.

Two quantizers are trained in parallel. Quantizer A uses $G = 4$ groups, each containing $K = 32$ embeddings at 50 frames-per-second. Quantizer B uses $G = 20$ groups. Ultra-low bitrate mode is achieved by calculating both quantizer outputs and adding the resulting embeddings. During training the quantizer B embedding is added with a probability of 50%.

IV. VOCODER

Following recent SOTA systems (Vocos [6], Wavehax [7]) we design a Sublime (SUB-band LInear Magnitude-phase Estimation) vocoder which converts the latent space of the quantizer z into a log-magnitude spectrogram $\hat{M}_{log}$ and raw phases $\hat{P}$ that are inverted using an $ISTFT$ ($20ms$ hop, $40ms$ window):

$$\hat{y} = ISTFT(e^{\hat{M}_{log} + i\hat{P}})$$

Input of the vocoder z is processed by 4 conv1d blocks (kernel size 3, 256 hidden channels, 384 feed-forward channels), then another 4 conv1d blocks (kernel size 3, 256 hidden channels, 256 feed-forward channels), then three separate sub-band conv2d decoders are used to produce three 4d tensors of shape $[batch, features, channels, time]$. All three tensors are concatenated along the channel dimension and projected via conv2d to a $[batch, 2, channels, time]$ tensor containing the log-magnitudes and phases. These sub-band decoders emit the following frequency bands: $[0 - 2kHz]$, $[2 - 6kHz]$ and $[6 - 12kHz]$. Each sub-band decoder is composed of a series of Pixel-Shuffle (PS) upsampling layers, each followed by Universal Inverted Bottleneck (UIB) block introduced in MobileNet V4 [2] and include multiplicative activation GEGLU [3]. PS layers upsample only in the channel dimension, however versions that upsample in time dimension coupled with 10ms or 5ms $ISTFT$ were also tested. The final configuration specifies 2 upsampling layers for the 1st and 2nd sub-band, each with upsample rate $[2, 1]$ and followed by a single UIB block with 8 feature maps, kernel size $[5, 3]$ in both depth-wise convolutions, and expansion factor 1.5. The last sub-band decoder uses a single upsample layer with upsample rate $[4, 1]$ and a single UIB block with kernel size $[3, 3]$.

Training of the vocoder utilized an ensemble of three discriminators: Multi-Period Discriminator (MPD), Multi-scale STFT Discriminator (MSSTFTD) and a Multi-scale Magnitude Discriminator (MSMAGD) which has the same architecture as MSSTFTD, but uses log-magnitude inputs, instead of complex-valued inputs. MSSTFT and MSMAGD use 128 feature maps.

V. TRAINING

Track 1 system has been trained in two phases. In both phases an encoder with a decoder has been both optimized with a waveform reconstruction task.

In the first phase, temperature has been annealed for $20k$ steps from an initial $T_0 = 0.02$ to $T_1 = 0.01$ with a cosine decay, then for additional $130k$ steps using an exponential decay, halving temperature every $8k$ steps. Temperature floor was set to $T_F = 0.01$. Losses used in this phase were multi-scale $L1$ mel loss with weight $w_{mel} = 10.0$, multi-scale $L1$ mfcc loss with weight $w_{mfcc} = 1.0$, as well as RWSS loss with weight $w_{rwss} = 1.0$, smoothing factor $\epsilon = 0.001$ and $q = 0.05$.

In the second phase, encoder was frozen, temperature set to $T = 0$ and discriminators were enabled. Training losses consisted of multi-scale $L1$ mel loss $w_{mel} = 10.0$, feature-matching loss $w_{fm} = 1.0$ and discriminator loss $w_d = 1.0$. In this phase the network was trained for a total of $160k$ steps which is short of the full convergence.

Track 2 system has been trained by freezing the Track 1 system, and training a separate frontend used instead of the encoder-SAVQ, with a cross-entropy objective. Prediction of the codes is assumed to be conditionally independent between the codebooks, and during inference greedy decoding is performed. Track 2 system has been trained for a total of $120k$ steps.

All three training runs use AdamW optimizer and follow a cosine learning-rate decay between $lr_0 = 2e-4$ and $lr_{200k} = 1e-4$, with effective batch size of 32.

VI. DATASET

In Track 1, first phase trained with the full provided training set, with a segment size of 3 seconds. Phase 2 trained with a clean split of the provided training set, with a segment size of 1 seconds. Track 2 system was trained with a clean split of the training set, using full utterances and batch zero-padding. Clean split was obtained by calculating UTMOS score and taking the top 60% of all utterances.

All training runs set the gain of audio to a $dB\ RMS$ level drawn randomly from a $[-18dB, -6dB]$ range. Inputs of the model are degraded by a sequence of data augmentation steps. First, random RIR from the provided set of RIRs is convolved with the input (with probability 25% for Track 1 and 40% for Track 2), then random noise from the provided set of training noises is added with a randomly sampled $dB\ SNR$ ($[6dB - 30dB]$ for Track 1 and $[-6dB - 30dB]$). Lastly a down-sampling is simulated with probability 20% of obtaining $8kHz$ sampling rate, and 50% of obtaining $16kHz$ sampling rate.

VII. EVALUATION

Final model checkpoint has been selected by comparing UTMOS scores calculated on an open testset set, combined with manual listening. Tables I, II, III, IV, V, VI contain UTMOS Results of a submitted checkpoint followed by results of a converged checkpoint (trained for a total of 1.3M steps for Track 1 and 2M steps for Track 2) in parentheses. All UTMOS values are calculated on an open testset.

Model	Clean
baseline (1 kbps)	1.44
proposed (1 kbps)	2.49 ± 0.5 (2.69 ± 0.51)
baseline (6 kbps)	3.23
proposed (6 kbps)	3.14 ± 0.57 (3.33 ± 0.57)

TABLE I
TRACK 1 UTMOS CLEAN

Model	Noisy
baseline (1 kbps)	1.33
proposed (1 kbps)	2.47 ± 0.47 (2.65 ± 0.48)
baseline (6 kbps)	2.7
proposed (6 kbps)	3.05 ± 0.54 (3.21 ± 0.54)

TABLE II
TRACK 1 UTMOS NOISY

Model	Reverb
baseline (1 kbps)	1.26
proposed (1 kbps)	2.16 ± 0.43 (2.28 ± 0.42)
baseline (6 kbps)	1.32
proposed (6 kbps)	2.58 ± 0.52 (2.7 ± 0.49)

TABLE III
TRACK 1 UTMOS REVERB

Model	Clean
baseline (1 kbps)	1.37
proposed (1 kbps)	2.48 ± 0.48 (2.69 ± 0.48)
baseline (6 kbps)	2.97
proposed (6 kbps)	3.14 ± 0.56 (3.36 ± 0.55)

TABLE IV
TRACK 2 UTMOS CLEAN

Model	Noisy
baseline (1 kbps)	1.35
proposed (1 kbps)	2.35 ± 0.51 (2.71 ± 0.53)
baseline (6 kbps)	2.56
proposed (6 kbps)	2.85 ± 0.6 (3.24 ± 0.56)

TABLE V
TRACK 2 UTMOS NOISY

Model	Reverb
baseline (1 kbps)	1.32
proposed (1 kbps)	2.27 ± 0.47 (2.63 ± 0.49)
baseline (6 kbps)	1.79
proposed (6 kbps)	2.66 ± 0.55 (3.23 ± 0.55)

TABLE VI
TRACK 2 UTMOS REVERB

REFERENCES

[1] Liu, Z., Mao, H., Wu, C., Feichtenhofer, C., Darrell, T. & Xie, S. A ConvNet for the 2020s. (2022), https://arxiv.org/abs/2201.03545

[2] Qin, D., Leichner, C., Delakis, M., Fornoni, M., Luo, S., Yang, F., Wang, W., Banbury, C., Ye, C., Akin, B., Aggarwal, V., Zhu, T., Moro, D. & Howard, A. MobileNetV4 – Universal Models for the Mobile Ecosystem. (2024), https://arxiv.org/abs/2404.10518
[3] Shazeer, N. GLU Variants Improve Transformer. (2020), https://arxiv.org/abs/2002.05202
[4] Zeghidour, N., Luebs, A., Omran, A., Skoglund, J. & Tagliasacchi, M. SoundStream: An End-to-End Neural Audio Codec. (2021), https://arxiv.org/abs/2107.03312
[5] Mentzer, F., Minnen, D., Agustsson, E. & Tschannen, M. Finite Scalar Quantization: VQ-VAE Made Simple. (2023), https://arxiv.org/abs/2309.15505
[6] Siuzdak, H. Vocos: Closing the gap between time-domain and Fourier-based neural vocoders for high-quality audio synthesis. (2024), https://arxiv.org/abs/2306.00814
[7] Yoneyama, R., Miyashita, A., Yamamoto, R. & Toda, T. Wavehax: Aliasing-Free Neural Waveform Synthesis Based on 2D Convolution and Harmonic Prior for Reliable Complex Spectrogram Estimation. (2024), https://arxiv.org/abs/2411.06807