EFFICIENT REAL-TIME AUDIO CODEC WITH INTEGRATED SPEECH ENHANCEMENT TECHNIQUES

Weihao Xiong, Congxin Zhang, Xinming Yan, Qingbo Huang

ByteDance

ABSTRACT

This paper presents our submission model for the 2025 Low-Resource Audio Codec (LRAC) Challenge, which is an efficient real-time audio codec with integrated speech enhancement techniques. The model is composed of three primary components: an encoder, a quantizer, and a decoder. To achieve better performance, the encoder module encodes the noisy audio into clean embeddings with the constraint of a pretrained codebook. Then the decoder decoders the clean embedding to audio wavforms. This system operates with a 50ms latency and a computational complexity of 2.68 GFLOPS, with the decoder contributing 0.58 GFLOPS.

***Index Terms*—** Speech Enhancement, audio codec

1. OVERVIEW OF OUR SYSTEM

This model is primarily built upon advancements from previous codecs and vocoders [1, 2, 3, 4]. The model operates in the frequency domain, where an STFT (Short-Time Fourier Transform) is applied before the encoder and an iSTFT (Inverse STFT) is performed after the decoder. Within the model, only the magnitude of the STFT is processed.

The STFT and iSTFT processes utilize a 50ms window length (1200 samples at 24000hz) and a 12.5ms hop size (300 samples), resulting in a total latency of 50ms. The 1kbps codebook consists of 5,792 numbers, producing a bitrate of 1kbps, calculated as $\frac{1}{12.5} \cdot \log_2(5792)$. For the 6kbps configuration, the model extends the 1kbps codebook with six additional codebooks, each containing 1,024 numbers. This setup results in a total bitrate of 5.8kbps, computed as $\frac{1}{12.5} \cdot (\log_2(5792) + \log_2(1024) * 6))$.

The model includes a total of 11.96 million trainable parameters and has a computational complexity of 1.34 GMacs (2.68 GFLOPS)."

2. ENCODER

The encoder module, referred to as the **FullBandEncoder**, processes input data through three main components: an input feature extractor (`in_fc`), a series of eight sequential blocks (`blocks`), and an output projection head (`out_head`). The overall architecture has **8.2M parameters** and a computational complexity of **662.66MMACs (Million Multiply-Accumulate operations)**, which accounts for **50.829% of the total parameters** and **39.592% of the total MACs** in the network. The encoder aims to capture temporal and spatial dependencies in sequential data while maintaining high computational efficiency.

2.1. Input Feature Extraction (`in_fc`)

The first component of the encoder is the input feature extractor (`in_fc`), which processes the raw input data. This module has **1.34M parameters** and contributes **108.31MMACs** (**8.294% Params, 6.471% MACs**). It consists of the following layers:

- **ChannelNormalization**: This normalization layer stabilizes the input data by re-scaling the channel distributions. As a computationally free module (*0% Params, 0% MACs*), it improves model training and convergence behavior.
- **Conv1d Sequential Block**:
  - **ConstantPad1d**: Padding is applied (**padding=(2, 0)**) to ensure dimensional alignment prior to convolution. This layer does not add any computational cost.
  - **Conv1d**: A convolutional layer with **1.34M parameters**, configured with **602 input channels** (in consistancy with stft freq bins), **740 output channels**, a **kernel size of 3**, and a **stride of 1**. This operation extracts local features while increasing dimensionality to match the hidden size.

2.2. Sequential Blocks (`blocks`)

Motivated by [5], the second component consists of eight **Large Kernel Convolution-Style Attention Blocks (LKCABs)**, which are implemented through a `ModuleList`. Each block has **833.98k parameters** and a computational complexity of **67.37MMACs** (**5.170% Params, 4.025%**

MACs). Collectively, the blocks account for the primary processing in the encoder.

Each `LKCAB` focuses on capturing temporal and spatial dependencies through its **attention module**, **value module**, and **output projection layer**. These are described in detail below:

2.2.1. Attention Module (`attn`)

The attention module processes sequential data through a combination of normalization, convolutional operations, and non-linear activations in the following pipeline:

- **ChannelNormalization**: A normalization layer prepares the input channels for processing without adding to the computational complexity (*0% Params, 0% MACs*).
- **First Conv1d Sequential Block**:
  - **ConstantPad1d**: Padding ensures consistent input dimensions without adding parameters or MACs.
  - **Conv1d**: This convolutional layer has **275.28k parameters** and operates on **740 input and output channels**. It uses a **kernel size of 1**, **stride of 1**, and **groups=2**, enabling separable convolution for efficient feature extraction. It accounts for **22.24MMACs**, or **1.706% Params, 1.329% MACs**.
- **GELU Activation**: A **GELU (Gaussian Error Linear Unit)** introduces non-linearity into the pipeline. GELU has no parameters and a negligible computational cost of **59.94KMACs** (**0.004% MACs**), but it provides smooth and continuous activation for improved gradient flow and feature learning.
- **Second Conv1d Sequential Block**:
  - **ConstantPad1d**: Padding aligns the input sequence for the subsequent convolutional layer.
  - **Conv1d**: A depthwise convolutional layer configured with **8.14k parameters**. It operates on **740 input and output channels**, with a **kernel size of 9**, **stride of 1**, and **groups=740**, allowing each channel to be processed independently. This layer consumes **599.4KMACs** (**0.050% Params, 0.036% MACs**) and captures channel-specific features over a larger receptive field.

The attention module has a total of **283.42k parameters** and contributes **22.9MMACs** (**1.757% Params, 1.368% MACs**). By combining separable convolutions, non-linear activation, and depthwise operations, it efficiently extracts both local and global features from sequential data.

2.2.2. Value Module (`v`)

The value module processes the input in parallel to the attention module and comprises:

- **ConstantPad1d**: Padding ensures dimensional consistency without adding computational cost (*0% Params, 0% MACs*).
- **Conv1d**: A convolutional layer with **275.28k parameters** configured identically to the first Conv1d layer in the attention module (**740 input and output channels**, **kernel size 1**, **stride 1**, **groups=2**). It contributes **22.24MMACs**, or **1.706% Params, 1.329% MACs**.

The combined output of the attention module and the value module is connected by a **residual connection**, ensuring stable training and strong information flow.

2.2.3. Projection Layer (`proj`)

The output projection layer refines features by projecting the channels back to the hidden dimensionality. This layer includes:

- **ConstantPad1d**: Padding is applied to preserve spatial consistency.
- **Conv1d**: Another convolutional layer with **275.28k parameters** identical to those in the value module. It contributes **22.24MMACs** (**1.706% Params, 1.329% MACs**).

2.3. Output Projection Head (`out_head`)

The final component of the encoder processes the output of the eight sequential blocks to produce the desired feature representation. The output projection head has **189.95k parameters** and contributes **15.37MMACs** (**1.177% Params, 0.918% MACs**). It consists of:

- **ConstantPad1d**: Padding ensures dimensional alignment.
- **Conv1d**: A convolutional layer with **740 input channels**, projecting down to **256 output channels**. It uses a **kernel size of 1** and **stride of 1**. This operation reduces dimensionality while retaining relevant features for downstream tasks.

2.4. Overall Design and Applications

The encoder module leverages **channel normalization**, **separable convolutions**, **depthwise operations**, and **residual connections** to achieve efficient and expressive feature extraction. Its modular design makes it suitable for sequential data tasks such as **audio signal processing**. By balancing computational complexity (**662.66MMACs**) and parameter

count (**8.2M**), the encoder strikes an excellent trade-off between performance and resource efficiency.

3. QUANTIZATION

As previously mentioned, both the 1kbps and 6kbps configurations share a base codebook consisting of 5,792 entries, which is quantized by a vector quantizer (FactorizedVectorQuantize) [6] with a computational complexity of 381.07 MMacs.

The 6kbps configuration introduces additional codebooks, which are quantized in two groups using another quantizer called SimVQ1D[7] with a computational complexity of approximately 1.2 MMacs.

The encoder always produces encodings at 6kbps, while in the training phase, the quantizer randomly drops the outputs from the second codebook. During inference, the decoder reconstructs the waveform using the specified codebook configuration.

4. DECODER

The decoder shares the same fundamental building blocks as the encoder but adopts two distinct configurations depending on the training stage. During stage 1, the hidden dimension is set to 600 to improve the performance of codebook training. In stage 2 and during inference, the hidden dimension is reduced to 530 to prioritize computational efficiency while maintaining strong performance.

To further address computational complexity constraints, the block dimensionality in the decoder is fixed at 530, and the number of blocks is limited to 6. This optimization reduces the computational complexity of the decoder to **296.96 MMACs**, meeting the competition's requirements while ensuring robust performance.

Apart from the hidden dimension and the number of blocks, the input dimension of the decoder is set to 256, which differs from that of the encoder.

For waveform reconstruction, the decoder incorporates multiple head modules to recover both the magnitude and phase components of the frequency-domain signal. These include:

- **Mag Head**: Outputs the magnitude of the STFT.
- **R Head and I Head**: Jointly output the phase information of the STFT.

Each head (Mag Head, R Head, and I Head) consists of a `ConstantPad1d` layer followed by a `Conv1d` layer. These modules share the same structure, with a 530-channel input, a 601-channel output, and a kernel size and stride of 1.

The final step in the decoder is the `ISTFT` (Inverse Short-Time Fourier Transform) layer, which reconstructs the time-domain signal from its frequency-domain representation. This layer introduces no additional trainable parameters or computational overhead, ensuring an efficient mapping back to the audio waveform.

5. TRAINING SETUP

We trained the model using data provided by the LRAC challenge requirements, with augmentation applied on the fly during training. The augmentation module simulates audio degradation by adding noise, reverberation, and other artifacts. It generates corresponding pairs of noisy and clean audio for training purposes.

The training process consists of two stages. In the first stage, the model is trained using clean speech as both input and output. During this phase, we aim to simultaneously learn the codebooks for both 1kbps and 6kbps configurations. In the second stage, the quantization module is frozen, and the encoder and decoder are retrained using noisy speech as the input and clean speech as the output.

This approach allows the codebooks to constrain the encoder to focus exclusively on encoding clean speech. On one hand, the denoising functionality is embedded within the encoder, which is a larger component of the model. On the other hand, the codebooks are specifically designed to support clean speech transmission, making them highly effective for this task.

Several loss functions are employed during the training process, including the multi-resolution STFT loss, multi-resolution Mel loss, and phase loss. Additionally, for adversarial loss, we utilize the Multi-Period Discriminator (MPD), Multi-Resolution STFT Discriminator (MRSTFTD), and Multi-Band Discriminator (MBD) to improve audio quality and realism.

6. REFERENCES

[1] Ilya Loshchilov and Frank Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations*, 2019.

[2] Rithesh Kumar, Prem Seetharaman, Alejandro Luebs, Ishaan Kumar, and Kundan Kumar, "High-fidelity audio compression with improved rvqgan," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[3] Andong Li, Tong Lei, Zhihang Sun, Rilin Chen, Erwei Yin, Xiaodong Li, and Chengshi Zheng, "Learning neural vocoder from range-null space decomposition," *arXiv preprint arXiv:2507.20731*, 2025.

[4] 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*, vol. 30, pp. 495–507, 2021.

[5] Ming-Ming Cheng Qibin Hou, Cheng-Ze Lu and Jiashi Feng, "Conv2former: A simple transformer-style convnet for visual recognition," in *IEEE Trans. Pattern Anal. Mach. Intell., vol. 46, no. 12, pp. 8274– 8283*, 2024.

[6] Chi-Min Chan Xinsheng Wang Xu Tan Jiahe Lei Yi Peng Haohe Liu Yizhu Jin Zheqi Dai Hongzhan Lin Jianyi Chen Xingjian Du Liumeng Xue Yunlin Chen Zhifei Li Lei Xie Qiuqiang Kong Yike Guo Wei Xue Zhen Ye, Xinfa Zhu, "Llasa: Scaling train-time and inference-time compute for llama-based speech synthesis," *arXiv preprint arXiv:2502.04128*, 2025.

[7] Yifei Xin Zhihua Xia Linli Xu Yongxin Zhu, Bocheng Li, "Addressing representation collapse in vector quantized models with one linear layer," *arXiv preprint arXiv:2411.02038*, 2025.