AUCOM: Extreme Compression for Real-Time Edge-to-Server Universal Audio Streaming

Yu Lu, *Student Member, IEEE,* Ran Wang, Dian Ding*, *Member, IEEE,* Yijie Li, Longyuan Ge, Juntao Zhou, *Student Member, IEEE,* Yongzhao Zhang, *Member, IEEE,* Yi-Chao Chen, *Member, IEEE,* Jiannong Cao, *Fellow, IEEE,* and Guangtao Xue, *Member, IEEE*

Abstract—Real-time audio streaming transmission and processing play a crucial role in time-sensitive applications such as food delivery services and ride-hailing platforms, where rapid response is essential. However, existing server-based audio streaming architectures struggle to handle the high concurrency of massive mobile devices efficiently. Traditional compression methods like MP3 and AAC offer limited compression ratios, while deep learning-based approaches often fail to meet the real-time transmission demands of edge computing environments. In this paper, we propose a novel edge-to-server audio streaming architecture that leverages Mel filter bank spectral features to achieve ultra-high compression efficiency. Our system integrates audio denoising, Mel feature extraction, and quantization-based compression at the edge, effectively suppressing environmental and device-induced noise while achieving an extreme compression ratio of 0.39% relative to the original uncompressed audio. Compared to conventional methods like MP3, our approach further reduces the file size by 96.1%. The decompressed Mel features remain task-independent, enabling seamless support for various general-purpose audio processing tasks in the server. We evaluate our system across three key audio tasks: speech recognition, speech emotion recognition, and audio classification. Extensive experiments on five different mobile devices demonstrate a 93.10% reduction in transmission latency at 1 Mbps bandwidth compared to 64 kbps MP3 audio, while maintaining task performance within a 5% deviation from state-of-the-art (SOTA) models across six mainstream audio datasets. These results highlight the efficiency, robustness, and scalability of our approach for real-time edge-to-server audio processing.

Index Terms—Mobile streaming audio application, Ambient noise adaptation, Mel-spectrogram-based audio compression

1 INTRODUCTION

To protect service quality and safety, companies such as Didi Dache and Meituan record the audio of delivery staff and drivers during the service process and upload it to the corporate server in real-time. The purpose is to record the live situation of the service process to protect the reasonable rights and interests of users and staff and to make timely responses to accidents such as car accidents and disputes. According to statistics [1], Didi Dache had 3.746 billion ride-sharing or meal delivery orders in the first quarter of 2024, creating a demand for real-time transmission, processing, and storage of massive amounts of audio.

The real-time transmission, processing, and storage of massive audio streams challenge the existing edge-to-server architecture. Firstly, although the audio is not stored directly on the driver's and passenger's mobile phones, the long duration of audio transmission will lead to traffic and energy consumption problems. In addition, as the scale of users increases, for example, the number of active drivers in China has reached 10 million [1]. The parallel transmission of large-scale mobile devices will increase the load on the platform, which may affect the stability of the system and the continuity of the service.

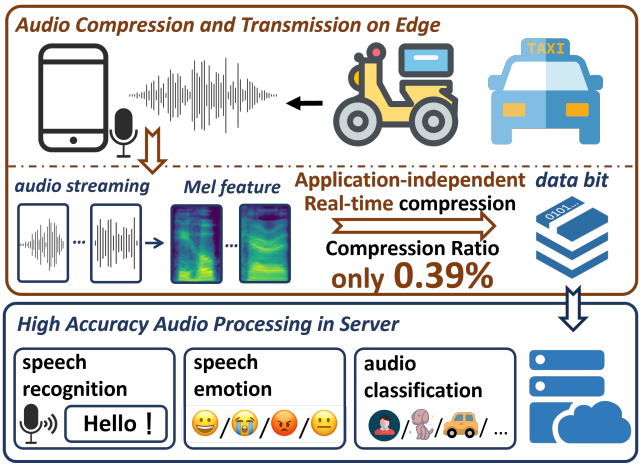


Fig. 1: The audio streaming architecture, AUCOM, enables efficient real-time audio compression and transmission on the edge. The compressed Mel features are independent of downstream tasks and can support universal audio processing tasks in the server.

The existing audio streaming architecture eases the burden by compressing audio files. The lossless audio compression techniques such as FLAC [2], ALAC [3], APE [4], and

* *Corresponding author.*

- *Y. Lu, R. Wang, D. Ding, L. Ge, J. Zhou, Y. Chen, and G. Xue are with the Department of Computer Science and Engineering, School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China. E-mail:{yulu01, wang_r, dingdian94, gly2000, juntaozhou, yichao, gt_xue}@ sjtu.edu.cn.*
- *Y. Li is with the School of Computing, National University of Singapore, 117417, Singapore. E-mail:yijieli@nus.edu.sg.*
- *Y. Zhang is with Department of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China. E-mail:zhangyongzhao@uestc.edu.cn.*
- *Jiannong Cao is with Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China, and also with Research Institute for Artificial Intelligence of Things (RIAIoT), The Hong Kong Polytechnic University, Hong Kong, China. E-mail: csjcao@comp.polyu.edu.hk.*

WAV [5] are capable of compressing audio files by $40-60\%$. MP3 [6], AAC [7], and other methods achieve compression rates of $10-20\%$ by removing some of the audio data that is imperceptible to the human ear. For example, when recording audio in MP3 format on mobile, the commonly used bit rate is $128kbps$, and recording 10 hours of audio will generate a file of about $562.5MB$. However, as the user scale continues to expand, the above methods are difficult to fundamentally solve the real-time transmission load in the case of high concurrency of massive mobile devices and the storage pressure caused by massive audio files.

Deep learning-based audio compression technology uses neural networks to build the Auto-Encoder for compression, significantly improving the compression ratio while ensuring audio quality [8], [9]. Encodec [9] compressed $24KHz$ audio down to $1.5kbps$ based on a multi-scale spectral adversarial network. DAC [8] introduced Residual Vector Quantisation (RVQ) based on Adversarial Generative Networks to compress $44.1KHz$ audio to $8kbps$. However, complex network structures such as the transformer [9] make it difficult to balance the computation and transmission burden of edge devices and introduce additional energy consumption and severe latency issues. Furthermore, the audio reconstructed by these networks at extremely low bit rates ($\leq 3kbps$) tends to exhibit significant distortion.

In response to the dilemma of current audio streaming architectures, we propose the following hypothesis: **Is it possible to achieve real-time audio compression and transmission with low computation and high compression rate on the edge, and achieve high-precision processing of generic downstream speech tasks in the server?**

In this paper, we propose a novel edge-to-server architecture for audio streaming based on a novel audio feature compression model, AUCOM. Inspired by task offloading [10], [11], the system attempts to implement real-time extraction and compression of audio features at the edge and use the compressed universal audio features in the server for tasks such as speech recognition [12], [13], speech emotion recognition [14], [15], [16], and audio classification [17], [18], [19]. The system breaks through the performance bottlenecks of existing audio streaming architectures and differs from previous systems in two main aspects:

- **Compression efficiency**: The system proposes a compression network based on rate-distortion optimization, which achieves real-time audio compression with a low compression rate and low computation on the mobile side. The audio processing and compression time for 1-minute audio is only $\sim 200ms$, and the compressed audio features are only 0.39% of the original audio.
- **Feature Universality**: Distinguished from task-specific feature extraction methods, the audio Mel filter bank features are independent of the downstream task. In this paper, we test three typical audio tasks: speech recognition, speech emotion recognition, and audio classification. The compressed features can be adapted to various SOTA models, and the model accuracy of multiple datasets is kept within 5% of the deviation from the original audio.

Building a real-time and efficient audio streaming architecture presents several challenges. One major issue is the presence of multiple noise interferences during audio acquisition on mobile devices. These interferences include environmental noise (e.g., traffic noise, crowd chatter, wind noise, and mechanical noise) as well as device-induced noise (e.g., circuit noise and electromagnetic interference), both of which can significantly degrade the quality of subsequent audio processing tasks.

To address this, we propose a deep learning-based denoising network, AUCOMSENET, which adopts a codec architecture to perform audio denoising and speech recovery in the time-frequency domain. The model leverages the Performer mechanism to efficiently capture temporal and spectral dependencies while applying power-law compression to the amplitude spectrum for improved denoising accuracy. Through parallel amplitude mask decoding and phase decoding, the network predicts refined amplitude and phase information, ultimately reconstructing high-quality audio signals. Furthermore, we incorporate a GAN-based training strategy with a metric discriminator to enhance optimization and improve overall denoising performance.

The second challenge lies in the inherent redundancy of audio data, which makes it difficult for traditional audio-based compression schemes to break through efficiency bottlenecks and achieve more effective compression. To address this, we propose a Mel-spectrogram-based compression approach, leveraging the compact and expressive nature of Mel features to maximize audio compression efficiency. Unlike conventional image compression, however, Mel feature compression requires a quantization model that simultaneously considers both frequency and temporal dimensions.

To this end, we introduce AuComNet, a neural network-based audio compression model built around an edge device to server communication pipeline that is architecture-agnostic: the "server" can be an edge node in a three-layer device–edge–cloud stack or a cloud node in a two-layer device–cloud setup. On the device, the compression module extracts fbank features and applies Generalized Divisive Normalization (GDN) to obtain compact representations; during training, we relax quantization via additive uniform noise to enable end-to-end differentiability, and at runtime we perform entropy coding for lossless bitstream generation. On the server side (edge or cloud server), the decompression module performs entropy decoding, inverse normalization, and parametric synthesis to reconstruct the fbank features. The decoder is modular and configurable, allowing integration with dynamic offloading policies that consider real-time network conditions, device capabilities, and server load; while a full policy design is beyond our scope, our design explicitly supports both two- and three-layer deployments.

Finally, to satisfy the requirements of downstream audio applications, the decompressed Mel features need to support the efficient execution of various audio tasks. To achieve this, we utilize the Whisper [20], UMNOS [21], and AST [22] models deployed in the server for performing speech recognition, speech emotion recognition, and audio classification tasks, respectively. The main contributions of this work are as follows:

- **Robust Audio Denoising**: We propose an audio denoising module based on a time-domain codec architecture, integrating a GAN-based training strategy to effectively reconstruct magnitude and phase information. This

enables efficient and high-fidelity audio denoising on edge devices.

- **Efficient Audio Compression**: We introduce a Mel-spectrogram-based compression scheme, implementing a Mel feature extraction and quantization compression model at the edge device. This approach overcomes the bottleneck of traditional audio compression, achieving an extreme compression ratio of 0.39%, which corresponds to a 96.1% reduction in file size compared to conventional methods (e.g., MP3) under similar task accuracy constraints.
- **Generalizable Audio Decompression**: We design a server-based decompression module that reconstructs bitstream-based Mel features while ensuring independence from downstream tasks. The system seamlessly supports a variety of audio applications, including speech recognition, speech emotion recognition, and audio classification, while maintaining compatibility with state-of-the-art (SOTA) models such as Whisper, UMNOS, and AST.
- **Real-Time Cloud-Edge Streaming Architecture**: We propose and evaluate a real-time audio streaming architecture that can leverage both cloud and edge servers, optimizing edge-to-server communication via a modular design. This architecture supports both two-layer and three-layer configurations, making it adaptable to different deployment scenarios. We demonstrate the effectiveness of this architecture across a server and five mobile devices, conducting experiments on six mainstream audio datasets for three speech-related tasks. Our results show that, compared to MP3 at 64 kbps, our system reduces file transfer time by 93.10% for 1-minute audio transmission over a 1 Mbps upload bandwidth, significantly improving transmission efficiency.

2 RELATED WORK

2.1 Audio Compression

2.1.1 Traditional Compression

Traditional audio compression methods are mainly divided into two categories. lossless compression methods such as FLAC [2], ALAC [3], APE [4], and WAV [5] retain all the information of the original audio, removing redundant data without losing sound quality. Lossy compression methods such as MP3 [6] and AAC [7] discard the frequency information that is not easily perceived by the human ear. Although there is a loss of sound quality, it provides an adequate listening experience.

2.1.2 DL-based Compression

With the neural network model, significant compression ratios can be achieved while maintaining high-quality audio results. DAC [8] introduced Residual Vector Quantisation (RVQ) based on Adversarial Generative Networks to achieve high-fidelity compression of various types of audio. Encodec [9] employs techniques such as improved vector quantization of residuals and frequency domain reconstruction loss to lower bit rates while effectively reducing distortion and artifacts.

2.2 Audio Application

2.2.1 Speech Emotion Recognition

Speech emotion recognition has been studied for multiple decades within both the machine learning and speech communities. In alignment with the prevailing research approach, scholars extract feature insights from audio data and subsequently employ these insights across a range of classifiers [14], [15], [16]. Much of the aforementioned works relied on context to furnish additional information for correcting and inferring emotional content extracted from the data. The mining and analysis of emotional information from single-sentence audio data can pose more significant challenges. Xu et al. [23] introduced an attention-based network designed for aligning textual and audio information, along with feature extraction. Delbrouck [21] et al. proposed a transformer-based joint-encoding model called UMNOS for single-sentence emotion recognition and sentiment analysis.

2.2.2 Automatic Speech Recognition

Recent advances in deep learning have greatly improved automatic speech recognition (ASR). Modern end-to-end ASR systems typically consist of an encoder that extracts high-level acoustic features from input audio, followed by a decoder that transforms these features into text sequences. End-to-end deep CNN models were initially explored in [24] and subsequently enhanced with depth-wise separable convolutions [25] and the Squeeze-and-Excitation module [26]. However, CNNs often struggle to capture global contexts effectively, transformer models [27] have been widely incorporated into backbone architectures due to their capability to capture long-range dependencies between speech frames [12], [13].

2.2.3 Audio Classification

Recent advancements in audio classification have demonstrated the effectiveness of various architectures and models. CNN architectures [17] are highly effective for large-scale audio classification tasks. Palanisamy [18] shows that ImageNet-pretrained deep CNN models are effective for audio classification. Causal Audio Transformer (CAT) [19] is designed specifically for audio classification with optimized features and a causal module. Zhu et al. [28] introduce the Multiscale Audio Spectrogram Transformer (MAST) to efficiently generate more semantically distinct feature representations.

3 PRELIMINARY STUDY

3.1 Mel-spectrogram Based Application

Mel Spectrum transforms the spectrum based on the Mel scale, converting the spectral information of an audio signal into a representation that is more compatible with the auditory properties of the human ear [29]. The spectral information in the high-frequency part is more aggregated, while the spectral information in the low-frequency part is more detailed. It is commonly used in speech recognition [30], [31], speech emotion recognition [32], [33], and audio classification [17], [18], [19]. The Mel spectrum is more compact and efficient than the original audio, thus we attempt to

explore the extreme compression of audio streaming based on the Mel fbank spectrum.

3.2 Mel-spectrogram Compression

In recent years, neural network-based compression methods [8], [9], [34], [35], [36] have emerged and been applied to compress various media files, such as images, audio, and video. Compared to traditional compression techniques such as JPEG, MP3, and H.264/AVC, these neural network-driven approaches have demonstrated significant improvements in both compression efficiency and reconstruction quality. Given that the Mel-spectrogram features of audio are represented as two-dimensional vectors analogous to images, we employ a lossy compression technique, akin to those used in image compression [37], [38], [39], to achieve an efficient compression of the Mel-spectrogram.

3.2.1 Quantization-based Compression

During the compression of Mel-spectrogram features, quantization is utilized to reduce the amount of information required for storage or transmission, though this process also introduces errors. Moreover, this quantization is not performed directly on the 2D vector of Mel-spectrogram features. Instead, a latent representation of that vector is identified, i.e., a vector r in another space, which is then quantized to produce the discrete vector $\hat{r}$. The discrete vector $\hat{r}$ can be losslessly compressed using entropy coding methods [29], [40], [41], generating a bitstream that can be stored or transmitted over a channel. Entropy coding relies on a priori probability models of the quantized representation (i.e., the entropy model for Mel-spectrogram coding), which are known to both the encoder and decoder of the entropy model.

3.2.2 Entropy model for Mel-spectrogram Coding

The entropy model for Mel-spectrogram coding is used to measure the redundancy of information of Mel-spectrogram's latent representation feature. For the latent vector r of the Mel-spectrogram, which consists of two dimensions, the feature dimension and the temporal dimension. Considering that the feature dimensions have a fixed length (as derived from the fixed feature dimensions of the Mel-spectrogram), we sequentially encode each feature dimension. We assume that each feature dimension j follows a continuous probability distribution P_j. All points on feature dimension j, $r_{j1}, r_{j2}, \ldots, r_{jn}$, are defined to belong to independent probability distributions $p_{j1}, p_{j2}, \ldots, p_{jn}$ (n is the width of the time dimension). As shown in Fig. 2, the probability distributions p_{ji} are all nearly Gaussian, so we have the following approximation:

$$p_{ji} \approx \mathcal{N}(\mu_{ji}, \sigma_{ji}^2) \tag{1}$$

P_j is a mixed probability distribution of p_{ji}:

$$P_j \approx \sum_{i=0}^{n} \frac{1}{n} \mathcal{N}(\mu_{ji}, \sigma_{ji}^2) \tag{2}$$

The entropy $H(r_j)$ of the latent vector r_j on feature dimension j is defined as:

$$H(r_j) = -\sum P_j(r_j) log_2 P_j(r_j) \tag{3}$$

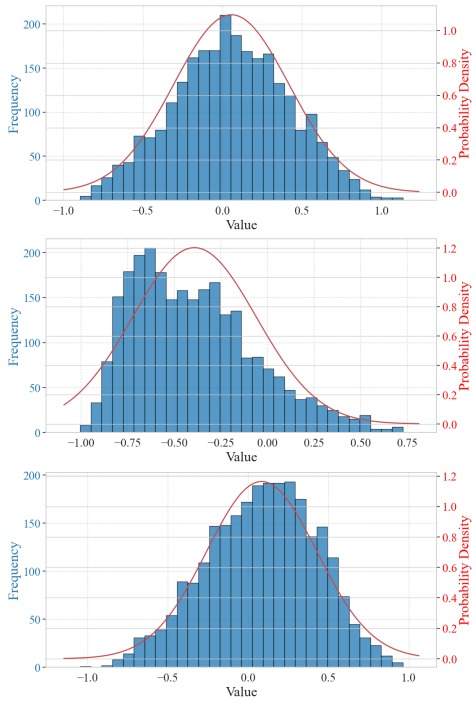


Fig. 2: The probability density function, statistically obtained from three points of r_{ji} on the dataset, approximates a Gaussian distribution. Each r_{ji} has a different probability density function approximating a Gaussian distribution $\mathcal{N}(\mu_{ji}, \sigma_{ji}^2)$.

According to the **Shannon–Fano coding theorem** or **Huffman coding** [42], the average code length L_a for encoding a symbol (in this case, a feature value r_{ji}) is related to its entropy. For optimal (prefix) codes, the average code length L_a is approximately equal to the entropy:

$$L_a \approx H(r_j) \tag{4}$$

This implies that when the entropy $H(r_j)$ is low (i.e., the feature values are highly predictable or redundant), the average code length will also be small. Conversely, a higher entropy, signifying less redundancy, results in a larger average code length. To minimize the average encoding length as much as possible, it is necessary to reduce the entropy of the feature values. An effective approach for achieving this is to normalize each variable along the temporal dimension:

$$\tilde{r}_{ji} = \frac{r_{ji} - \mu_{ji}}{\sigma_{ji}}, \tilde{r}_{ji} \sim \mathcal{N}(0,1) \tag{5}$$

The normalized mixed probability distribution and the entropy are as follows:

$$\tilde{P}_j \approx \mathcal{N}(0,1), \tilde{H}(\tilde{r}_j) = -\sum \tilde{P}_j(\tilde{r}_j) log_2 \tilde{P}_j(\tilde{r}_j) << H(r_j) \tag{6}$$

Subsequently, we quantize the normalized latent features $\tilde{r}$ to obtain $\hat{r}$ and employ entropy coding methods to achieve an average encoding length L_a that approximates the entropy.

4 AUCOM ARCHITECTURE

4.1 System Overview

The system overview of AUCOM, as illustrated in Fig. 3, outlines a edge-to-server audio streaming architecture that

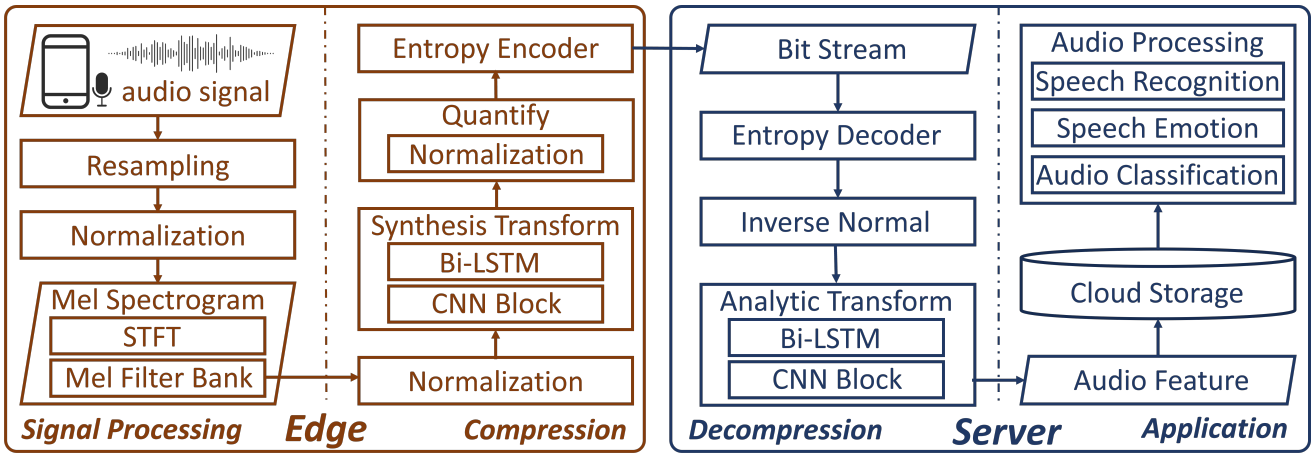


Fig. 3: The cloud-edge audio streaming architecture of AUCOM

efficiently processes, compresses, and transmits audio signals from the edge to the server for various applications. The architecture is divided into four main stages: Signal Processing, Compression, Decompression, and Application:

- Signal Processing (Edge): On edge devices, audio signals captured via microphones undergo preprocessing (e.g., resampling, normalization). In noisy conditions, signals are processed through a speech enhancement module for noise reduction before Mel-frequency filter bank (Mel-Fbank) feature extraction. In clean environments, signals bypass enhancement, enabling adaptive, resource-efficient feature extraction based on noise levels.
- Compression (Edge): In this stage, the normalized features are compressed using a series of transformations on the edge device. An analytic transform, incorporating Bi-LSTM networks and CNN blocks is applied. The resulting data is further normalized, quantified, and encoded by an entropy encoder to generate a compact bit stream.
- Decompression (Server): The bit stream is transmitted to the server, where it is decoded using an entropy decoder. The decompressed data undergoes an inverse normalization and is subsequently subjected to a synthesis transform using Bi-LSTM and CNN blocks to recover the Mel fbank spectral features.
- Application ((Server): The decompressed audio features are then utilized for various applications such as speech recognition, speech emotion analysis, and audio classification. Additionally, the audio features can be stored in the server for further processing.

4.2 Audio Signal Processing

After recording with the microphone on the edge, AUCOM first process the audio. Considering that most mobile applications on edge devices use a 44.1 kHz sample rate as the default configuration to record audio, but that a 16 kHz sample rate is sufficient for most audio application scenarios, we resample the 44.1 kHz audio signal to 16 kHz. This approach ensures audio quality while reducing the discrete Fourier transform's computational load by a factor of approximately $2.76\times$ floating-point operations [43]. The audio signal is then normalized, typically to ensure a consistent amplitude range, often between -1 and 1.

4.2.1 Audio Enhancement

AUCOM use microphones to capture audio information and convert it to a discrete digital signal. However, after recording, the audio may contain background noise and microphone self-noise, which can degrade its quality. To address this, we apply audio enhancement techniques to suppress noise and improve the signal-to-noise ratio (SNR) [44], resulting in higher-quality audio. This, in turn, enhances the accuracy of downstream tasks.

Existing audio enhancement methods can be broadly categorized into two classes: time-domain and time-frequency (TF) domain approaches. Time-domain audio enhancement methods [45], [46], [47], [48], [49] leverage neural networks to learn a mapping from noisy waveforms to clean waveforms. However, these methods face challenges in effectively utilizing spectral information, which can lead to increased signal distortion. Additionally, they exhibit weak long-term dependency modeling, limited capability in handling non-stationary noise, high computational costs, and poor generalization to unseen noise conditions. In contrast, time-frequency (TF) domain audio enhancement methods have demonstrated superior performance. These methods aim to predict clean frame-level TF representations and subsequently reconstruct the enhanced waveform, effectively leveraging spectral information for improved noise suppression and audio quality.

Time-frequency domain audio enhancement methods convert the audio signal into time-frequency features using the Short-Time Fourier Transform (STFT). After processing these features, the enhanced audio signal is then reconstructed, enabling noise reduction and overall audio enhancement. For example, due to the phase spectrum's wrapping and non-structured nature, enhancing it is challenging. Thus, some approaches [50], [51], [52], [53] focus solely on magnitude spectrum enhancement, reconstructing the waveform using the Inverse Short-Time Fourier Transform (ISTFT) with the enhanced magnitude and the original noisy phase. Meanwhile, other approaches [54], [55], [56], [57] bypass this issue by directly enhancing the real and imaginary parts of the time-frequency features in the complex do-

main, thereby reconstructing high-quality audio. However, enhancing both magnitude and phase directly provides a more accurate and natural signal reconstruction, improves audio quality by preserving the temporal structure, ensures phase integrity, and offers a more consistent enhancement compared to separately enhancing the real and imaginary parts in the time-frequency domain [58]. Consequently, similar to works [59], [60], [61], [62], [63], we convert the audio signal into a time-frequency representation using the STFT, perform denoising on both the magnitude and phase spectra simultaneously, and then reconstruct the enhanced speech signal with improved quality.

Next, we will introduce our proposed specific network model AUCOMSENET and training methods.

4.2.1.1 AUCOMSENET: As shown in Fig. 4, we provide an overview of the proposed AUCOMSENET model architecture. The AUCOMSENET network uses an encoder-decoder structure to denoise noisy audio $A_n \in \mathcal{R}^L$ in the TF domain and recover the clean speech signal $A_c \in \mathcal{R}^L$ where L is the signal length. Specifically, we first apply STFT to transform the audio signal into the TF domain, obtaining the magnitude spectrum $X_M \in \mathbb{R}^{T_x \times F_x}$ and phase spectrum $X_P \in \mathbb{R}^{T_x \times F_x}$, where T_x and F_x represent the total number of frames and frequency bins, respectively. To achieve more accurate denoising of the magnitude spectrum, we apply power-law compression, resulting in the compressed magnitude spectrum $\hat{X}_M = X_M^r$, where $r \in [0, 1]$ is 0.3 by default. Next, we concatenate X_M and X_P to obtain $X_{in} \in \mathcal{R}^{2 \times T_x \times F_x}$, and use the encoder E_{ae} to extract the time-frequency domain representation from X_{in}. In the encoder E_{ae}, we employ the Performer [64] architecture to progressively capture both temporal and spectral dependencies at each stage. Finally, we simultaneously predict the magnitude spectrum mask M_x and the clean phase spectrum $\tilde{X}_P$ using parallel magnitude mask and phase decoders. The clean magnitude spectrum is obtained as $\tilde{X}_M = (M_x \cdot \hat{X}_M)^{\frac{1}{r}}$ and, together with the clean phase spectrum $\tilde{X}_P$, is used to reconstruct the high-quality audio signal $\tilde{A}_c$. In addition, we employ a GAN-based [65] approach to train our model, utilizing a metric discriminator to assist in the training process. Next, we provide a detailed introduction to the architecture of the encoder and decoder in our network model.

4.2.1.2 Details of AUCOMSENET Encoder.: As shown in Fig. 5, our encoder module E_{ae} extracts effective representations $E_{ae}(X_{in}) \in \mathcal{R}^{N_x \times T_x \times \frac{F_x}{2}}$ from the input X_{in} in the TF domain (N_x is 64 by default). The network architecture is centered around the Wavelet Transform Dense Block (WTDenseBlock), designed to effectively capture local and multi-scale features that are crucial for representing complex audio TF characteristics. Specifically, the Encoder consists of the following components:

Initial Convolutional Layer: The layer consists of a convolutional block to expand the feature dimensionality of N_x, followed by instance normalization [66] for training stability and a PReLU activation [67] to introduce non-linearity, thereby enhancing the model's ability to capture complex patterns.

Wavelet-Transformed Dense Block (WTDenseBlock): As shown in Fig. 7, the WTDenseBlock integrates dense connections [68] with wavelet-based convolutions [69] to enhance hierarchical feature reuse and multi-resolution analysis. Each layer begins with a convolutional block to capture spatial dependencies, followed by a GELU activation function to introduce non-linearity. Subsequently, a wavelet transform convolution (WTConv2d) is applied to extract multi-scale features, effectively expanding the receptive field and improving both the model's robustness and computational efficiency. Instance normalization is employed to stabilize the training process, while PReLU activation enhances model adaptability. For the i-th layer of the WTDenseBlock, the forward propagation process is defined as follows: The forward pass is:

$$x_{i+1} = PReLU(IN(WTConv2d(GeLU(Conv2d(x_i))))) \tag{7}$$

The output x_{i+1} of the i-th layer is concatenated with its input x_i, and the combined feature map is passed to the $(i+1)$-th layer, enhancing information flow, promoting feature reuse, and mitigating the vanishing gradient problem.

Downsampling Convolutional Layer: This layer first applies a convolutional block to reduce the frequency dimension of the features, with a default downsampling factor of 2. This is followed by instance normalization and PReLU activation to obtain the output $X_{in}^d \in \mathcal{R}^{N_x \times T_x \times \frac{F_x}{2}}$.

Times Performer and Frequency Performer. The input tensor $X_{in}^d \in \mathcal{R}^{N_x \times T_x \times \frac{F_x}{2}}$ is reshaped for temporal attention using the PerformerBlock [64], which operates with 4 attention heads and an input dimensionality of 64. The resulting features are integrated via a residual connection to enhance temporal representation. The output is then reshaped for frequency-domain attention, again leveraging the same PerformerBlock configuration, and combined with the input through another residual connection to reinforce frequency feature learning. Finally, the tensor is reshaped to its original dimensions, $N_x \times T_x \times \frac{F_x}{2}$. This dual-attention mechanism effectively captures temporal and spectral dependencies, improving the model's ability to handle sequential and frequency variations.

4.2.1.3 Details of AUCOMSENET Magnitude Mask Decoder.: As illustrated in Fig. 6, The magnitude mask decoder predicts the magnitude mask M_x from the TF domain representation and multiplies it with the noisy magnitude spectrum $\hat{X}_M$ to obtain the clean magnitude spectrum $\tilde{X}_M$. For the input $E_{ae}(X_in)$ from the encoder, the magnitude mask module first applies a WTDenseBlock to extract rich hierarchical features. Subsequently, transposed convolution is employed to upsample the feature map, restoring the original frequency dimension. This process helps recover detailed spectral information lost during the downsampling operations in the encoder. Next are instance normalization and the PReLU layer After these operations, a final convolutional layer is used to reduce the feature dimension from N_x to 1, effectively condensing the multi-channel feature representations into a single-channel magnitude mask $\hat{M}_x$. To achieve precise and adaptive magnitude mask prediction, we further adopt a learnable sigmoid (LSigmoid) [70] activation function, defined as:

$$M_x = \frac{W}{1 + e^{1 - b\hat{M}_x}} \tag{8}$$

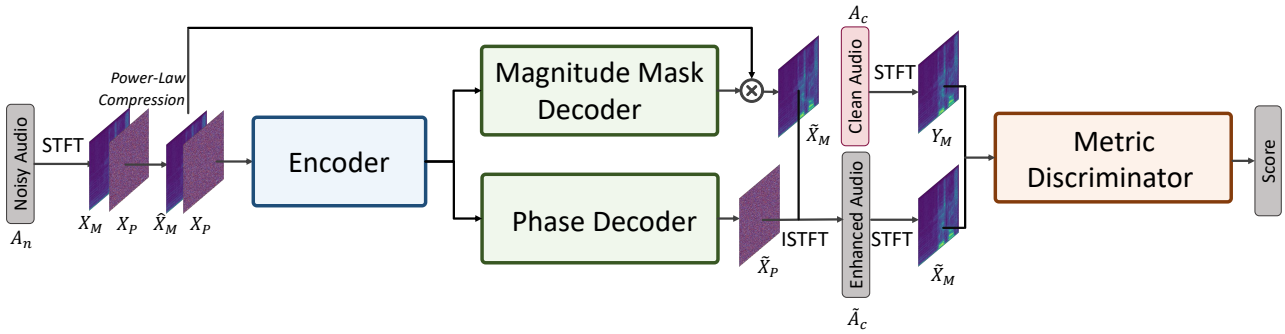


Fig. 4: Detailed framework of our audio enhancement model AUCOMSENET

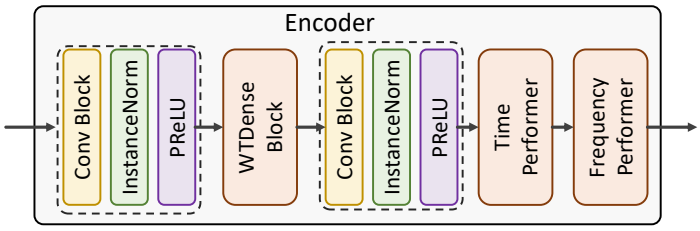


Fig. 5: AUCOMSENET Encoder

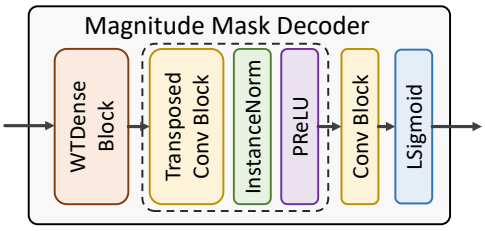


Fig. 6: Magnitude Mask Decoder

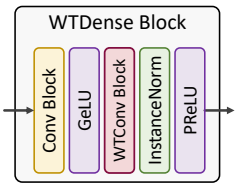


Fig. 7: WTDense Block

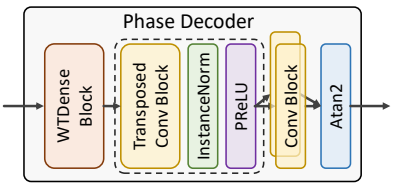


Fig. 8: Phase Decoder

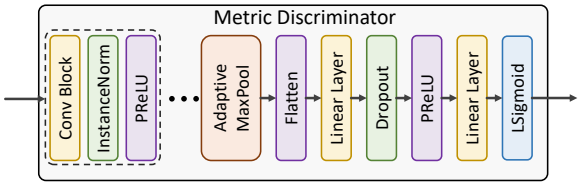


Fig. 9: AUCOMSENET Metric Discriminator.

where $W = 2.0$ and $b \in \mathcal{R}^{1\times F}$ are learnable parameters. Unlike the standard sigmoid function, LSigmoid introduces learnable parameters that allow the model to adjust the non-linearity dynamically, thereby improving its capacity to capture subtle variations in the audio signal.

4.2.1.4 Details of AUCOMSENET Phase Decoder.: As illustrated in Fig. 8, the phase decoder directly predicts the clean phase spectrum $\tilde{X}_P$ from the TF domain representation. For the input $E_{ae}(X_i n)$ from the encoder, the phase decoder first applies a WTDenseBlock to extract rich hierarchical time-frequency features. Following feature extraction, two separate convolutional blocks are employed to predict the real part $\tilde{X}_P^r$ and the imaginary part $\tilde{X}_P^i$ of the phase spectrum, respectively. This separation allows the network to model the distinct characteristics of each component more effectively, enabling better representation of the underlying phase information. To reconstruct the clean phase spectrum, we apply a bi-parametric arctangent function (atan2), defined as:

$$\begin{aligned}\tilde{X}_P &= atan2(\tilde{X}_P^i, \tilde{X}_P^r) \\ &= \arctan\left(\frac{\tilde{X}_P^i}{\tilde{X}_P^r}\right) + \pi \cdot \frac{1-\text{sgn}^{\S}(\tilde{X}_P^r)}{2} \cdot \text{sgn}^{\S}(\tilde{X}_P^i)\end{aligned} \tag{9}$$

where $\text{sgn}(x)^{\S}$ is a redefined function which equals to 1 when $x \geq 0$, and equals to -1 when $x < 0$. The $atan2$ function computes the angle of the complex number formed by X_p^r and X_p^i, ensuring accurate phase estimation across the full range of $-\pi$ to π. This formulation provides robustness in handling phase discontinuities and improves the phase reconstruction quality. Modeling real and imaginary components with the $atan2$ function, the phase decoder improves phase estimation, enabling high-fidelity audio reconstruction in noisy environments.

4.2.1.5 Details of AUCOMSENET Metric Discriminator.: The Metric Discriminator evaluates feature similarity between magnitude spectrum pairs, as shown in Fig. 9, predicting a similarity score aligned with the scaled PESQ (Perceptual Evaluation of Speech Quality in ITU-T Recommendation P.862. Unlike conventional distortion measures such as MSE or SNR, PESQ is a model-based evaluation algorithm without a simple closed-form mathematical expression. Instead, it consists of a multi-stage processing pipeline, including signal preprocessing, temporal alignment via time warping, frequency domain transformation using perceptual Bark-scale filters, perceptual distortion modeling, and nonlinear mapping to generate a final quality score ranging from -0.5 to 4.5.) [71] metric. It consists of four convolutional blocks with instance normalization and PReLU activation for stable, hierarchical feature learning. An adaptive max-pooling layer reduces spatial dimensions, followed by flattening and two spectral normalized linear layers. A dropout layer (0.3 rate) after the first linear layer prevents overfitting, with PReLU introducing non-linearity. Finally, a learnable sigmoid activation outputs a similarity score between 0 and 1.

4.2.1.6 Training Strategies: We adopt a GAN-based approach to train the model. First, we train the discrimina-

tor, which is designed to evaluate the perceptual quality of the audio. We rescale the PESQ scores to the range of (0, 1) and use them as target values to train the discriminator. The discriminator takes pairs of clean and predicted enhanced magnitude spectra (Respectively represented as Y_M and $\tilde{X}_M$) as input and is trained to output the scaled PESQ scores, effectively learning to assess the perceptual quality of the enhanced audio:

$$\begin{aligned}\mathcal{L}_{MD} = &||MD(\tilde{X}_M, Y_M) - Q_{pesq}(A_c, sg[\tilde{A}_c])||_2^2 \\ &+ ||MD(Y_M, Y_M) - 1||_2^2\end{aligned} \tag{10}$$

where $sg[\cdot]$ is the stop gradient operator, MD denotes the metric discriminator and $Q_{pesq} = \frac{PESQ(\cdot)-1}{3.5}$ denotes the scaled PESQ score and $PESQ$ [71] is an objective metric used to evaluate the quality of speech signals.

Next, we train the generator. To measure the difference between the generated audio and the clean audio, we utilize a combination of loss functions, including the magnitude spectrum difference, the time-domain waveform difference, and the complex-valued difference in the TF domain:

$$\begin{aligned}\mathcal{L}_{time} = ||A_c - \tilde{A}_c||_1, \mathcal{L}_{mag} = ||\tilde{X}_M - Y_M||_2^2 \\ \mathcal{L}_{complex} = ||STFT(A_c).real - STFT(\tilde{A}_c).real||_2^2 \\ +||STFT(A_c).imag - STFT(\tilde{A}_c).imag||_2^2\end{aligned} \tag{11}$$

These complementary loss functions help the generator learn to produce high-quality audio that closely resembles the clean reference, both in spectral content and temporal structure. Subsequently, we leverage the discriminator to generate a metric loss:

$$\mathcal{L}_{metric} = ||MD(\tilde{X}_M, Y_M) - 1||_2^2 \tag{12}$$

This adversarial training process encourages the generator to optimize not only for signal fidelity but also for perceptual quality.

Finally, consistent with the anti-wrapping loss [63], [72], we define three phase-specific loss functions to optimize the phase spectrum: the instantaneous phase loss, the group delay loss, and the instantaneous angular frequency loss. The core component of these losses is the anti-wrapping function, defined as

$$\mathcal{F}_{AWL}(x) = |x - round(\frac{x}{2\pi} \cdot 2\pi)| \tag{13}$$

This function effectively maps phase differences to the principal range, mitigating the effects of phase discontinuities. The instantaneous phase loss measures the direct phase difference between the predicted phase and the ground truth phase:

$$\mathcal{L}_{ip} = ||\mathcal{F}_{AWL}(\tilde{X}_M - Y_M)||_1 \tag{14}$$

By applying the anti-wrapping function, this loss penalizes large phase discrepancies, ensuring phase alignment across the dataset. The group delay loss evaluates the difference in temporal phase gradients (first-order differences along the time dimension) between the predicted and ground truth phases:

$$\mathcal{L}_{gd} = ||\mathcal{F}_{AWL}(\triangle_{DT}(\tilde{X}_M) - \triangle_{DT}(Y_M))||_1 \tag{15}$$

where $\triangle_{DT}$ represents the first-order difference along the time dimension, measuring the rate of change in phase over time. This loss promotes temporal smoothness and consistency, crucial for time-dependent applications. The instantaneous angular frequency loss focuses on frequency-domain phase gradients, computed as first-order differences along the frequency dimension:

$$\mathcal{L}_{iaf} = ||\mathcal{F}_{AWL}(\triangle_{DF}(\tilde{X}_M) - \triangle_{DF}(Y_M))||_1 \tag{16}$$

where $\triangle_{DF}$ represents the first-order difference along the frequency dimension, capturing the rate of change in phase across different frequency bins. This loss helps preserve spectral coherence, ensuring that the frequency characteristics of the predicted output align with the ground truth.

Therefore, the final loss function for the generator is formulated as a weighted sum of the aforementioned loss components, integrating magnitude, time-domain, complex-valued, metric, and phase-specific losses. The overall objective function is defined as:

$$\begin{aligned}\mathcal{L}_{total} = &\lambda_1\mathcal{L}_{mag} + \lambda_2\mathcal{L}_{time} + \\ &\lambda_3\mathcal{L}_{complex} + \lambda_4\mathcal{L}_{metric} + \lambda_5\mathcal{L}_{phase\_total}\end{aligned} \tag{17}$$

where

$$\mathcal{L}_{phase\_total} = \mathcal{L}_{gd} + \mathcal{L}_{ip} + \mathcal{L}_{iaf} \tag{18}$$

the coefficients $\lambda_1, \lambda_2, \lambda_3, \lambda_4$, and λ_5 are hyperparameters that control the relative contributions of each loss term and are set to (0.9,0.2,0.1,0.05 and 0.3) by default, allowing for flexible adjustment to balance signal fidelity, perceptual quality, and phase consistency in the generated audio.

4.2.2 Mel Fbank Feature Extraction

We then extract the Mel filter bank spectrogram features from the audio. Beginning with framing and windowing, the audio signal is divided into overlapping frames, and each frame is multiplied by a window function, such as a Hamming or Hanning window, to minimize spectral leakage which occurs when signal discontinuities distort the frequency spectrum. Next, the framed and windowed signal undergoes the STFT, which converts the signal from the time domain to the frequency domain, resulting in a spectrogram that represents the signal's frequency content over time. The spectrogram is then transformed into a Mel spectrogram by applying a set of Mel filter banks that compress the frequency axis to a Mel scale, a scale that approximates the way humans perceive pitch. Finally, log scaling is applied to the Mel spectrogram to reduce the dynamic range, which enhances the perceptual loudness of the sounds and makes the quieter parts of the audio more distinguishable. After the process described above, for standard mobile device audio (typically with a 44.1 kHz sample rate, 16-bit bit depth, and dual channels), the filterbank converts it into a matrix of size $H \times L$, where H is typically 80 or 128 for higher frequency resolution, L is $100 \times T$, and T is the duration of the audio (default unit is s). This matrix represents a Mel filter bank feature.

4.3 Compression Model

In this section, we introduce the neural network-based compression model, AUCOMNET. As shown in Fig. 10, the compression model consists of two parts: the compression module on the edge and the decompression module on the

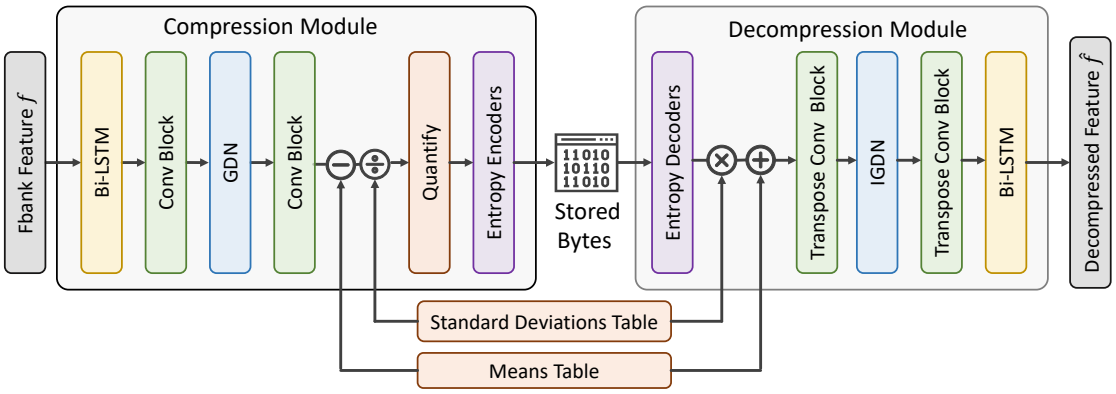


Fig. 10: Detailed framework of our compression model AuComNet

server. Similar to the transform coding approach to image compression [73], the encoder employs stacked nonlinear neural network modules to transform the fbank features f into potential representations r. Subsequently, r is normalized and quantized into its discrete form $\hat{r}$. This value can then be losslessly compressed using entropy coding techniques such as arithmetic coding [74] and Huffman coding, and stored or transmitted as a sequence of bits. Then, the decompression module recovers $\hat{r}$ from the compressed signal sequence and applies inverse normalization and parametric synthesis transform to reconstruct the fbank feature $\hat{f}$.

Next, we provide a detailed description of the model.

4.3.1 Compression Module on Edge

As mentioned in the previous Sec. 3.2, the compression module deployed on the edge mainly consists of the parametric analytic transform function h_f, the quantization operator $\mathcal{Q}$, and the entropy encoders. For the input fbank feature $f \in \mathcal{R}^{1\times H\times L}$, h_f is first used for feature extraction via a bidirectional LSTM structure [75]. Immediately thereafter, the features are processed and downsampled using a convolutional neural network. Meanwhile, a non-linear transformation called Generalized Divisive Normalization (GDN) [76] is applied between convolutional blocks. The primary function of the GDN module is to apply a form of local normalization that helps the network learn more efficient and compact representations of data. Eventually, h_f outputs the feature $r = h_f(f;\theta_f) \in \mathcal{R}^{M\times\frac{H}{4}\times\frac{L}{4}}$ (θ_f represents the optimizable parameters of the function h_f and M represents the number of output channels in the final convolutional block, with a default value of 2).

As mentioned in the previous Sec. 3.2, we normalize r before the quantization operator rounds r for quantization. We dynamically record the mean and variance of each point-independent probability distribution using the means table and the standard deviations table. After network optimization, the final values are calculated on the training set and saved. Finally, we encode the discrete matrix $\hat{r} \in \mathcal{R}^{M\times\frac{H}{4}\times\frac{L}{4}}$ using an entropy encoder. We first reshape $\hat{r}$ to the form $\frac{HM}{4}\times\frac{L}{4}$, i.e., two dimensions: feature dimension and temporal dimension. Each feature dimension is then encoded by **Huffman coding** [41]. The encoding algorithm is independent of model training and is applied only during the actual use of the model. We provide a detailed description of it in Sec. 4.3.4.

4.3.2 Decompression Module on Server

The decompression module, deployed on the server, is structured symmetrically to the compression module. It includes the entropy decoders and the parametric synthesis transform h_b. The entropy decoder decodes the byte stream into a discrete matrix $\hat{r}$, which is then inverse normalized, reshaped as $M\times\frac{H}{4}\times\frac{L}{4}$, and input into the transformation h_b. h_b first up-samples $\hat{r}$ using an inverse convolutional neural network containing the IGDN module. The Inverse Generalized Divisive Normalization (IGDN) is the inverse operation of the Generalized Divisive Normalization (GDN) and essentially reverses the normalization effect applied by GDN, helping to reconstruct the original signal from its compressed form. A bidirectional LSTM structure is then used to process the upsampled features and reconstruct them to obtain the Mel-spectrogram feature $\hat{f} = h_b(\hat{r};\theta_b)$ (θ_b represents the optimizable parameters of the function h_b).

4.3.3 Rate-Distortion Optimization Approach for Compression Model

The compression module introduces information distortion, with the quantization process in particular introducing error. Distortion refers to the difference between the reconstructed Mel-spectrogram features $\hat{f}$ and the original features f, which is quantified using the squared l_2-norm as follows:

$$\mathcal{L}_D = \sum_i (\hat{f}_i - f_i)^2 \tag{19}$$

This distortion is tolerable within lossy compression schemes and is expected to exhibit a negative correlation with the compression bit rate, meaning that as the bit rate decreases, the error increases. We define the expected encoding length of the compressed features as $\mathcal{R}$ (i.e., the bit rate). Based on the discussion in Sec. 3.2, assuming the entropy coding technique is executed effectively, the bit rate $\mathcal{R}$ can be expressed as the cross-entropy:

$$\mathcal{R} \approx \mathbf{E}_{f\sim P_f}[-log_2 P_{\hat{r}}(\hat{r})] = \mathbf{E}_{f\sim P_f}[-log_2 P_{\hat{r}}(\mathcal{Q}(\tilde{r}))] \tag{20}$$

It is worth noting that during training, we dynamically record the mean and variance of the latent representation r

across feature dimensions using learnable parameters, and normalize it to obtain $\tilde{r}$.

For the optimization of the compression model, our goal is to find an optimal parametric analysis transformation function $h_f(\cdot;\theta_f)$, and a parametric synthesis transformation function $h_b(\cdot;\theta_b)$, to balance the estimated bit rate $\mathcal{R}$ and reconstruction distortion $\mathcal{L}_D$. Therefore, we optimize the compression model using gradient descent, with the loss function defined as follows:

$$\mathcal{L} = \mathcal{R} + \lambda \mathcal{L}_D \tag{21}$$

where λ is the rate-controlling hyper-parameter.

To enable the use of gradient descent methods for optimizing the model's performance with the transform parameters θ_f and θ_b, the optimization problem must be relaxed. This relaxation is necessary because quantization often results in gradients for θ_f being nearly zero. Considering that the investigated approximations include replacing the gradient of the quantizer [77] or using additive uniform noise in place of the quantizer during training [76], we adopt the latter approach, reverting to actual quantization when the model is applied outside of the training state. That is, during model training, the quantization operator does not round the latent representation r but instead applies uniform noise to it. However, when the model is being tested or used, the quantization operator rounds the latent representation r:

$$\hat{r} = \mathcal{Q}(\tilde{r}) = \begin{cases} \tilde{r} + noise, training = True \\ round(\tilde{r}), training = False \end{cases} \tag{22}$$

where $noise \sim \mathcal{U}(-0.5, 0.5)$ by default. After applying uniform noise to the latent representation, we incorporate the noise into the prior model as well:

$$\begin{aligned} P_{\hat{r}} &= (P * \mathcal{U}(-0.5, 0.5))(\hat{r}) \\ &= \int_{-\infty}^{\infty} P(\tilde{r}) \mathcal{U}(\hat{r} - \tilde{r}| - 0.5, 0.5) d\tilde{r} \\ &= \int_{\hat{r}-0.5}^{\hat{r}+0.5} P(\tilde{r}) d\tilde{r} \\ &= \mathcal{F}(\hat{r} + 0.5) - \mathcal{F}(\hat{r} - 0.5) \end{aligned} \tag{23}$$

where $\mathcal{F}(\cdot)$ is the cumulative of the underlying density model. We use a parametric model to approximate the cumulative density function $\mathcal{F}_j(\cdot)$ of different feature dimensions j:

$$\begin{aligned} f_j^m &= A^{m-1} f_j^{m-1} + B^{m-1} \\ &\quad + C^{m-1} \odot tanh(A^{m-1} f_j^{m-1} + B^{m-1}) \\ \mathcal{F}_j &= Sigmoid(A^M f_j^M + B^M) \end{aligned} \tag{24}$$

where A, B, C are all learnable parameters, m represents the number of layers in the parametric model, with $1 \leq m \leq M$, and $f_j^0(\cdot)$ is the identity function. At this point, we can optimize the compression model using gradient descent.

4.3.4 Huffman coding

Huffman coding is a lossless data compression algorithm that assigns variable-length codes to input characters, with shorter codes assigned to more frequent characters. After completing model training, we transformed all the data in the training set into the corresponding latent representation r and calculated its mean and variance at each position for normalization. After completing the normalization, we calculate the probability density function for each coded channel based on the training set data and build a Huffman tree. Note that we need to record both the minimum and maximum values for each channel and constrain the upper and lower limits of the test data before applying Huffman coding. Additionally, to achieve a smaller coding bit rate, we introduce an extra parameter τ, which performs an additional division operation on the normalized latent vector $\tilde{r}$ to obtain $\frac{\tilde{r}}{\tau}$ before quantification and encoding. As τ increases, the coding bit rate decreases. However, this also leads to greater information loss, resulting in an increased error in recovering the feature $\hat{f}$.

5 EVALUATION

5.1 Experimental Setup

We implemented the AUCOM system on several Android phones (i.e., Xiaomi 11 Pro, vivo X80, Huawei nova 12, Google Pixel 6 pro, and Honor X40 GT), as well as on a server equipped with an NVIDIA Tesla A800 and 20 Mbps bandwidth.

5.1.1 Datasets and evaluation metrics

For various audio applications, we use the following datasets to evaluate the specific performance of AUCOM system.

5.1.1.1 Audio Enhancement.: We evaluate the audio enhancement module's performance using both objective metrics and perceptual measures on the VoiceBank+DEMAND dataset [78]. **VoiceBank+DEMAND**: This benchmark combines clean speech from the Voice Bank Corpus with real-world noise from the DEMAND dataset, featuring 28 speakers (14 male, 14 female), with 10 for training and 2 unseen speakers for testing. Noisy speech is generated by mixing clean utterances with various environmental noises at SNRs of 0, 5, 10, 15 dB (training) and 2.5, 7.5, 12.5, 17.5 dB (testing). We assess our AUCOMSENET model using PESQ, CSIG, CBAK, COVL, SSNR, and STOI metrics. **PESQ (Perceptual Evaluation of Speech Quality) [71]**: An objective metric simulating human auditory perception to compare clean and degraded speech quality. **CSIG (Signal Distortion Ratio)**: An objective metric predicting subjective signal distortion, which evaluates how closely enhanced speech resembles clean speech in terms of perceived signal distortion reduction. **CBAK (Background Noise Intrusiveness)**: An objective metric predicting subjective background noise intrusiveness, assessing the effectiveness of noise reduction while preserving speech clarity. **COVL (Overall Speech Quality)**: An objective metric predicting overall subjective speech quality, combining signal distortion and background noise suppression to reflect naturalness and intelligibility. **SSNR (Segmental Signal-to-Noise Ratio)**: An objective metric calculating SNR for short segments to assess detailed noise suppression performance. **STOI (Short-Time Objective Intelligibility)**: An objective metric measuring speech intelligibility in noisy conditions by comparing clean and processed signals in time-frequency domains.

5.1.1.2 Automatic Speech Recognition: **LibriSpeech** [31]: A benchmark dataset for speech recognition, containing around 1,000 hours of English audiobook recordings from LibriVox, with high-quality segmented audio and corresponding transcriptions for ASR system training and evaluation. **LJSpeech** [30]: A popular ASR dataset with 24 hours of high-quality recordings from a single female speaker reading non-fiction texts. It contains 13,100 short audio clips paired with transcriptions, valued for its consistency in speaker and recording quality.

Word Error Rate (WER) is a standard ASR accuracy metric, measuring transcription errors—substitutions, insertions, and deletions—relative to the reference transcription:

$$WER = \frac{Substitutions + Insertions + Deletions}{Total\,Words\,in\,Reference} \quad (25)$$

As detailed in Sec. 4.2, we converted audio from both datasets into Mel filter bank spectrograms and jointly trained the compression model. During testing, the Whisper model evaluates WER on each dataset's test set across different compression rates.

5.1.1.3 Speech Emotion Recognition: **IEMOCAP:** Interactive Emotional Dyadic Motion Capture (IEMOCAP) [32] is a widely used multimodal dataset for emotion recognition, containing 12 hours of audio-visual data, including speech, facial expressions, and body gestures. For speech emotion recognition, we utilize only the audio component. **VCEMO** [33] is a large-scale Chinese corpus for single-sentence emotion recognition, featuring diverse voiceprints and text from over 100 speakers. It captures real-world speech with varied accents and language features. We tested the system's accuracy in recognizing four emotions (i.e., angry, happy/excited, sad, and neutral) using IEMOCAP and VCEMO.

5.1.1.4 Audio Classification: **Audio Set** [79] is a large-scale dataset for audio event detection and classification, containing over 2 million 10-second clips from YouTube labeled with 500+ audio event classes, including speech, music, animal sounds, and environmental noises. It serves as a benchmark for audio classification tasks. The AST model was pre-trained on the AudioSet dataset and used to evaluate the AUCOM system's audio classification accuracy on the following two datasets: **ESC-50:** [80] ESC-50 is a dataset of 2,000 environmental audio clips, organized into 50 classes with 40 clips each. Each 5-second clip is sourced from public sound databases, offering diverse ambient sounds, making it widely used for audio classification tasks. **Speech Commands:** [81] Speech Commands by Google contains thousands of 1-second audio clips of 35 common words (e.g., "yes," "no," "stop") recorded by various speakers. It serves as a standard benchmark for audio classification accuracy.

5.2 Implementation Details

For our AUCOMSENET, all audio samples are resampled to 16 kHz. The input features are extracted from the raw waveform using the STFT, with the number of FFT points, Hanning window length, and hop size configured to 400, 400 samples (25 ms), and 100 samples (6.25 ms), respectively. Model training is conducted using the AdamW optimizer [82] for a maximum of 100 epochs, with an initial learning rate of 0.0005, which is reduced by a factor of 0.5 every 30 epochs to facilitate convergence.

Meanwhile, in our experiments of AUCOMNET, we utilized the Adam optimizer [83] to train the model for 100 epochs on an NVIDIA Tesla A800 GPU. The Adam optimizer is chosen due to its adaptive learning rate mechanism, which computes individual learning rates for different parameters. Specifically, the optimizer was initialized with a learning rate of $1e-4$, with $\beta_1 = 0.9, \beta_2 = 0.999$ and $\epsilon = 1e-8$. To enhance the convergence and performance of the model, we employed a learning rate scheduler, specifically the MultiStepLR scheduler [84]. The scheduler decreases the learning rate by a factor of 0.1 at predefined epochs, specifically at the 40th and 80th epochs. The batch size of the training phase is 64, and audio data is divided into $5s$ chunks as inputs.

We modified the convolutional block parameter M of AuComNet and the hidden layer size H_s of the Bi-LSTM to construct three models with different configurations: AuComNet-v1 ($M = 2, H_s = \frac{H}{2}$), AuComNet-v2 ($M = 1, H_s = \frac{H}{2}$), and AuComNet-v3 ($M = 1, H_s = \frac{H}{4}$), each corresponding to a different target bit rate. The AUCOM system selects the appropriate AuComNet variant based on the target compression rate, ensuring efficient execution of audio tasks.

For on-device evaluation, we export the trained PyTorch [85] encoder to ONNX [86] format and deploy them in an Android Studio [87] environment. The models are executed using the default inference backend without enabling GPU acceleration, i.e., the computations are performed solely on the CPU. This design choice is intentional, as our goal is to develop a general-purpose audio compression system that remains functional and efficient on edge devices without dedicated GPUs. By evaluating under CPU-only conditions, we ensure broader hardware compatibility and provide a conservative estimate of real-world latency performance. Moreover, the decoder-side modules, including neural reconstruction and downstream task models, are offloaded to the server and executed using GPU acceleration for real-time performance.

5.2.1 Baseline Systems of Audio Compression

We compare our system with seven other benchmark systems of audio compression, including three edge-to-server compression architectures, two traditional audio stream compression schemes, and two emerging compression methods based on complex neural networks.

- **DeepCOD [88]**: DeepCOD employs a lightweight convolutional block for fast feature downsampling on edge devices, followed by quantization and encoding. A self-attention-based reconstruction network on the server restores features with high accuracy. This design balances edge efficiency with server computation, making it ideal for resource-constrained devices and optimized for tasks like speech recognition.
- **Intp**: The interpolation algorithm performs downsampling on the Mel-spectrogram features at the edge,

and on the server side, bilinear or bicubic interpolation is applied. Subsequently, the downsampled Mel-spectrogram features are quantized and encoded.

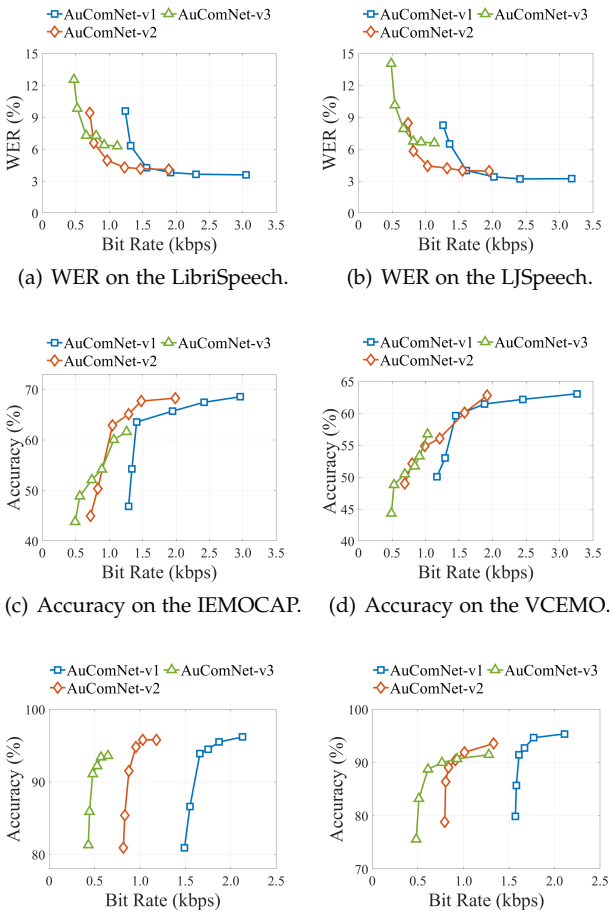
- **CS**: This method leverages compressed sensing for the compression and reconstruction of Mel-spectrogram features. We use ISTA-Net++ [89] to learn observation matrices for different compression rates, as well as a reconstruction network based on the ISTA algorithm.
- **MP3**: As a traditional audio compression method, MP3 uses psychoacoustic models to remove audio information that is inaudible to the human ear, such as quiet sounds or masked frequencies. It then applies techniques like framing, Discrete Cosine Transform (DCT), and quantization to compress the audio data while maintaining perceptual audio quality.
- **AAC**: AAC improves on MP3 through advanced psychoacoustic modeling, wider frequency support, and techniques like temporal noise shaping and prediction. These enhancements enable better audio quality and higher compression efficiency at lower bitrates.
- **EnCodec [9]**: EnCodec employs a streaming encoder-decoder framework with a multiscale spectrogram adversary to reduce artifacts and enhance audio quality. It uses a Transformer-based model to further compress the quantized latent space end-to-end, achieving much lower bitrates than traditional codecs like MP3 and AAC.
- **DAC [8]**: DAC compresses audio at 8 kbps by integrating high-fidelity audio generation with image-domain vector quantization and improved adversarial and reconstruction losses. Its universal architecture supports diverse audio types—speech, environmental sounds, and music—making it a versatile solution for general-purpose audio compression and generation.

5.3 Micro Benchmark

5.3.1 Micro benchmark of audio enhancement

We first evaluate the performance of our audio enhancement module using six mainstream objective audio quality assessment metrics, including PESQ, CSIG, CBAK, COVL, SSNR and STOI as previously described. Here, CSIG, CBAK, and COVL are objective metrics that predict subjective perceptual quality based on trained perceptual models, while PESQ, SSNR, and STOI directly measure perceptual quality, signal-to-noise ratio, and intelligibility, respectively. The evaluation is conducted on the VoiceBank + DEMAND dataset, where noisy signals are generated by adding noise recordings from the DEMAND dataset to clean speech from the VoiceBank corpus at varying signal-to-noise ratios (SNRs). For all the metrics, the evaluation is performed between the denoised audio and the clean ground-truth signal and higher values indicate better performance. To comprehensively assess the effectiveness of our AUCOMSENET, we compare it against 14 state-of-the-art audio enhancement approaches, such as SEGAN [45], MossFormer [56], TridentSE [92], and MP-SENet [63].

The results in Tab. 1 clearly demonstrate that our proposed method, AUCOMSENET, outperforms all existing audio enhancement approaches across key evaluation metrics. Specifically, our model achieves the highest scores in PESQ (3.54), CSIG (4.75), CBAK (3.99), and COVL (4.27), indicating superior performance in perceptual speech quality, signal clarity, background noise suppression, and overall audio quality. Additionally, our method maintains a highly competitive SSNR (10.83) and STOI (0.96), matching or exceeding the performance of other state-of-the-art models. These results highlight the robustness of our model in enhancing speech quality under various noisy conditions.



(a) WER on the LibriSpeech. (b) WER on the LJSpeech.

(c) Accuracy on the IEMOCAP. (d) Accuracy on the VCEMO.

(e) Accuracy on the Speech Commands. (f) Accuracy on the ESC-50.

Fig. 11: Analysis of AUCOM test performance metrics across varying bit rates for multiple datasets.

5.3.2 Micro benchmark of audio applications

As shown in Fig. 11, to assess the performance of various AUCOMNET, as well as provide a reference for the AUCOM system to select models for different target bit rates, a series of micro benchmark experiments were conducted across various datasets. For each model version (v1, v2, v3), we evaluate the bit rate using Huffman coding at $\tau \in [1, 2, 3, 5, 8, 10]$, alongside the corresponding metrics for the downstream tasks.

Speech Recognition: The evaluation focused on the specific metrics mentioned earlier: WER and accuracy, measured at different bit rates. Given that the WER of Whisper on uncompressed or lossless data (the bit rates are greater than 256 kbps) of LibriSpeech and LJSpeech is 3.58% and 3.04%, respectively, the results in Fig. 11(a) and Fig. 11(b) demonstrate that AUCOM achieves comparable performance at significantly lower bit rates (around 1.5

Method	Year	Input	PESQ↑	CSIG↑	CBAK↑	COVL↑	SSNR↑	STOI↑
Noisy	-	-	1.97	3.35	2.44	2.63	1.68	0.91
SEGAN [45]	2017	Waveform	2.16	3.48	2.94	2.80	7.73	0.92
DEMUCS [47]	2021	Waveform	3.07	4.31	3.40	3.63	-	0.95
SE-Conformer [48]	2021	Waveform	3.13	4.45	3.55	3.82	-	0.95
MossFormer [56]	2023	Waveform	3.47	4.40	3.50	3.73	9.09	**0.96**
MossFormer2 [57]	2024	Waveform	3.16	4.14	3.32	3.58	6.86	0.95
MetricGAN [90]	2019	Magnitude	2.86	3.99	3.18	3.42	-	-
MetricGAN+ [70]	2021	Magnitude	3.15	4.14	3.16	3.64	-	-
DPT-FSNet [91]	2021	Complex	3.33	4.58	3.72	4.00	-	**0.96**
FRCRN [55]	2022	Complex	3.23	4.29	3.47	3.83	7.60	0.95
TridentSE [92]	2023	Complex	3.47	4.70	3.81	4.10	-	0.96
DB-AIAT [93]	2021	Magnitude+Complex	3.31	4.61	3.75	3.96	10.79	-
CMGAN [59]	2022	Magnitude+Complex	3.41	4.63	3.94	4.12	**11.10**	**0.96**
PHASEN [62]	2020	Magnitude+Phase	2.99	4.21	3.55	3.62	10.18	-
MP-SENet [63]	2023	Magnitude+Phase	3.50	4.73	3.95	4.22	10.64	**0.96**
AUCOMSENET	2025	Magnitude+Phase	**3.54**	**4.75**	**3.99**	**4.27**	10.83	**0.96**

TABLE 1: Performance comparison of different audio enhancement methods.

kbps). Furthermore, AUCOM maintains a perfectly acceptable WER of less than 5% even at a bit rate of 1 kbps.

Speech Emotion Recognition: UMNOS achieved accuracies of 70.41% and 63.27% on the uncompressed or lossless test sets of IEMOCAP and VCEMO, respectively. As shown in Fig. 11(c) and Fig. 11(d), even at a bit rate as low as 1.5 kbps, the AUCOM system achieves near-lossless accuracy of 67.69% and 60.09%. When the bit rate is reduced to 1 kbps, the system still delivers acceptable accuracy: 62.89% and 56.72%.

Audio Classification: the AST model achieves accuracy rates of 96.55% and 95.52% on the Speech Command (Fig. 11(e)) and ESC-50 datasets (Fig. 11(f)), respectively, under uncompressed or lossless conditions. On the Speech Command dataset, AUCOM consistently achieves near-optimal accuracy at bit rates exceeding 1 kbps. For the ESC-50 dataset, AUCOM similarly achieved performance levels approximating those of the uncompressed or lossless case at bit rates exceeding 1.5 kbps, with accuracy remaining above 90% even at a bit rate as low as 1 kbps. Furthermore, the above results indicate that, compared to uncompressed or lossless compression methods (typically at 256 kbps), AUCOM can achieve effective compression of the original data at a bit rate close to 1 kbps, reducing the size to 0.39% of the original, with an acceptable accuracy loss of approximately 5%.

Fig. 12 illustrates the decompressed fbank spectrogram at various bit rates. It is evident that, even at a bit rate of 0.5 kbps, the decompressed spectrogram maintains its essential features remarkably well when compared to the uncompressed spectrogram. The micro benchmark results highlight the robustness and efficiency of our AUCOM system across various bit rates and datasets.

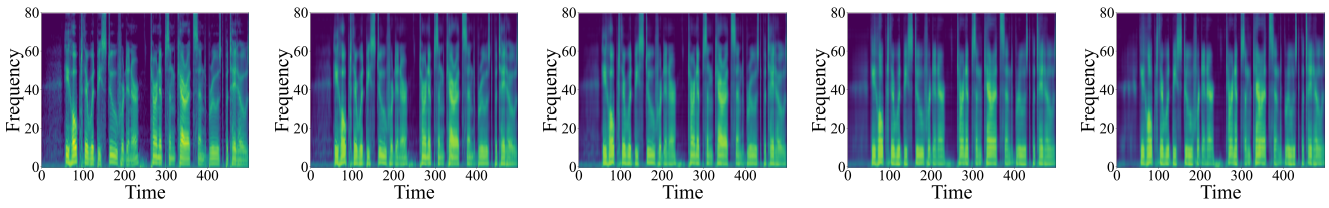
5.3.3 Impact of different entropy coding method

Additionally, we evaluated the performance of various entropy coding methods across different models and datasets, including Huffman coding [41]($\tau = 2$), Arithmetic coding [94], Range coding [95], and rANS coding [40]. Specifically, Arithmetic coding, Range coding, and rANS coding are applied to encode the same latent feature $\tilde{r}$, while Huffman coding is applied to encode $\tilde{r}/\tau$. The results in Tab. 2 indicate that, given the similar encoding efficiencies of all methods to the information entropy, the choice of encoding scheme does not significantly impact the system's compression rate or accuracy. Moreover, in the context of Mel-spectrogram encoding, a substantial amount of known features is available, enabling an approximate estimation of the feature's probability distribution. Given the relatively low computational overhead of Huffman coding and its efficiency (i.e., fast encoding/decoding speed) in compressing data with a fixed or near-fixed probability distribution, Huffman coding is particularly well-suited for the deployment of the AUCOM system on mobile edge devices.

5.4 Compared with other compression systems

We evaluate the trade-off between compression distortion and bit rate for AUCOM and various other compression baseline systems through an automated speech recognition task. Specifically, we tested the performance of different compression systems at various bit rates on the LibriSpeech dataset, with the results shown in Fig. 13.

Traditional algorithms such as MP3 and AAC exhibit a marked increase in WER as the bit rate is reduced, underscoring the significant degradation in speech recognition performance at very low bit rates. This decline suggests that these conventional methods are less effective in preserving speech intelligibility under constrained bit rate conditions. Although MP3 and AAC are built upon psychoacoustic models designed for perceptual fidelity, their optimization targets are not aligned with downstream tasks such as ASR. While certain parameters (e.g., quantization level, masking strength) can be tuned, their internal pipelines are based on fixed heuristics and are not differentiable or adaptable to task-specific objectives. In contrast, AuCom enables end-to-end training directly guided by downstream performance, offering better adaptability under extreme bitrate constraints. The edge-to-server offloading systems DeepCOD [88], CS, and Intp achieve lower compression bit rates compared to traditional algorithms while maintaining a low WER. This advantage is particularly pronounced in the neural network-based CS and DeepCOD systems. Among them, the ISTANet-based [89] CS method achieves a WER of less than 5% even at a bit rate close to 5 kbps. However, as the bit rate is further reduced (i.e., below 5 kbps), the error in these methods increases significantly. In contrast, the deep learning-based audio stream compression methods DAC [8] and EnCodec [9] demonstrate improved



(a) Uncompressed (b) ∼ 3kbps (c) ∼ 2kbps (d) ∼ 1kbps (e) ∼ 0.5kbps

Fig. 12: Comparison of decompressed fbank spectrogram at various bit rates and in the uncompressed state.

Model	AuComNet-v1				AuComNet-v2				AuComNet-v3			
Dataset	LibriSpeech		LJSpeech		LibriSpeech		LJSpeech		LibriSpeech		LJSpeech	
Metrics	Bit Rate ↓	WER ↓	Bit Rate ↓	WER ↓	Bit Rate ↓	WER ↓	Bit Rate ↓	WER ↓	Bit Rate ↓	WER ↓	Bit Rate ↓	WER ↓
Huffman	2.30 kbps	3.64%	2.41 kbps	3.20%	1.47 kbps	4.16%	1.55 kbps	3.98%	0.93 kbps	6.39%	0.93 kbps	6.66%
Arithmetic	3.42 kbps	3.58%	2.74 kbps	3.27%	2.01 kbps	4.06%	1.54 kbps	3.89%	1.32 kbps	6.23%	1.05 kbps	6.60%
Range	3.23 kbps	3.58%	2.53 kbps	3.27%	1.84 kbps	4.06%	1.53 kbps	3.89%	1.19 kbps	6.23%	0.98 kbps	6.60%
rANs	3.16 kbps	3.58%	2.19 kbps	3.27%	1.97 kbps	4.06%	1.41 kbps	3.89%	1.22 kbps	6.23%	0.93 kbps	6.60%
Dataset	IEMOCAP		VCEMO		IEMOCAP		VCEMO		IEMOCAP		VCEMO	
Metrics	Bit Rate↓	Acc ↑	Bit Rate↓	Acc ↑	Bit Rate↓	Acc ↓	Bit Rate↓	Acc ↑	Bit Rate↓	Acc ↑	Bit Rate↓	Acc ↑
Huffman	2.42 kbps	67.45%	2.45 kbps	62.18%	1.48 kbps	67.69%	1.58 kbps	60.09%	1.07 kbps	60.03%	0.91 kbps	53.28%
Arithmetic	2.92 kbps	70.19%	3.27 kbps	63.21%	1.97 kbps	68.52%	2.11 kbps	62.04%	1.49 kbps	59.14%	1.17 kbps	55.83%
Range	2.85 kbps	70.19%	3.08 kbps	63.21%	2.01 kbps	68.52%	1.99 kbps	62.04%	1.43 kbps	59.14%	1.05 kbps	55.83%
rANs	2.83 kbps	70.19%	3.19 kbps	63.21%	1.93 kbps	68.52%	2.04 kbps	62.04%	1.38 kbps	59.14%	1.06 kbps	55.83%
Dataset	Speech Commands		ESC-50		Speech Commands		ESC-50		Speech Commands		ESC-50	
Metrics	Bit Rate↓	Acc ↑	Bit Rate↓	Acc ↑	Bit Rate↓	Acc ↓	Bit Rate↓	Acc ↑	Bit Rate↓	Acc ↑	Bit Rate↓	Acc ↑
Huffman	1.87 kbps	95.50%	1.77 kbps	94.68%	1.03 kbps	95.81%	1.01 kbps	91.92%	0.57 kbps	93.42%	0.93 kbps	90.69%
Arithmetic	1.73 kbps	96.11%	2.16 kbps	95.10%	1.44 kbps	95.59%	1.97 kbps	94.96%	0.92 kbps	93.71%	1.73 kbps	93.92%
Range	1.61 kbps	96.11%	1.87 kbps	95.10%	1.31 kbps	95.59%	1.84 kbps	94.96%	0.83 kbps	93.71%	1.39 kbps	93.92%
rANs	1.59 kbps	96.11%	1.98 kbps	95.10%	1.04 kbps	95.59%	1.67 kbps	94.96%	0.69 kbps	93.71%	1.48 kbps	93.92%

TABLE 2: Impact of different entropy coding method

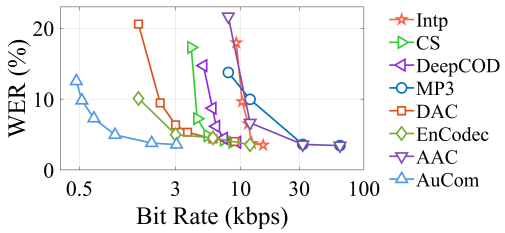


Fig. 13: Comparative analysis of WER across different compression systems on the LibriSpeech dataset.

Method	Parameter Number	FLOPs
DeepCOD	5.1e-2k	21.96M
CS	1.60M	36.52M
Intp	0M	20.64M
Encodec	7.43M	48.47G
DAC	19.36M	661.38G
AuCom (Ours)	65.00k	2.56G

TABLE 3: Computational complexity comparison of AuCom and representative baselines in terms of model size and FLOPs for 1-minute audio compression.

adaptability to bitrate reduction. However, as the bitrate approaches 2 kbps, their error rates still increase significantly. Moreover, compared to the edge-to-server offloading systems and traditional methods, these audio stream compression methods rely heavily on the computational power of servers, making them challenging to deploy at the edge. As a novel edge-to-server offloading system, AUCOM demonstrates outstanding performance across all tested bit rates, particularly excelling at extremely low bit rates. For instance, it achieves a WER of 4.95% at a bitrate of 0.97 kbps, a performance that surpasses all other systems. This demonstrates that, even under stringent bitrate constraints, AUCOM can effectively compress audio data to reduce transmission and storage overhead, while maintaining high speech recognition accuracy.

In addition to recognition performance, we further evaluate the computational efficiency of AUCOM compared with several representative baselines. Specifically, we compare the number of parameters and the total floating-point operations (FLOPs) required to compress 1-minute audio at 16 kHz using the encoder module of each method. The results are summarized in Tab. 3. For traditional codecs such as MP3 and AAC, it is difficult to report FLOPs due to their handcrafted signal processing pipelines and non-neural architecture. Therefore, we focus our comparison on other compression frameworks, including DeepCOD, CS, Intp, Encodec, and DAC.

As shown in Tab. 3, AuCom achieves a total of 2.56 GFLOPs for compressing 1-minute audio, which is significantly lower than Encodec (48.47 GFLOPs) and DAC (661.38 GFLOPs). With only 65K parameters, AuCom maintains a minimal model footprint, further validating its suitability for deployment on resource-constrained edge devices. The FLOPs of AUCOM are one to two orders of magnitude lower than other neural audio compression methods. This highlights the practicality of AuCom for real-time, low-power applications in bandwidth-limited mobile scenarios.

5.5 Performance on Mobile Deployment

5.5.1 *Micro benchmark tests on the test smartphones*

We deploy the signal processing and compression modules of the AUCOM system on edge devices, specifically mobile smartphones (i.e., 1: Xiaomi 11 Pro, 2: vivo X80, 3: Huawei nova 12, 4: Google Pixel 6 pro, and 5: Honor X40 GT). We first provide the detailed specifications of the mobile devices used in our evaluation, including the chipset model, memory size (RAM), and Android OS version. These details are listed in Tab. 4 to ensure the reproducibility

and clarity of our experimental setup. We then evaluate the performance of the smartphones using the ANTUTU benchmark suite [96], which covers five primary indicators: CPU, GPU, MEM, UX, and AI. The results are shown in Tab. 4. Specifically, CPU, GPU, and MEM denote the mobile phone's CPU performance, 3D performance, and RAM performance, respectively, while the UX indicator integrates data security, data processing, image processing, and I/O performance. Meanwhile, the AI benchmark test evaluates the performance of a device's artificial intelligence capabilities, including tasks like image processing, natural language understanding, and real-time AI applications. It provides a comprehensive score reflecting the efficiency and power of AI operations on the device.

5.5.2 Evaluation of system deployment on mobile devices.

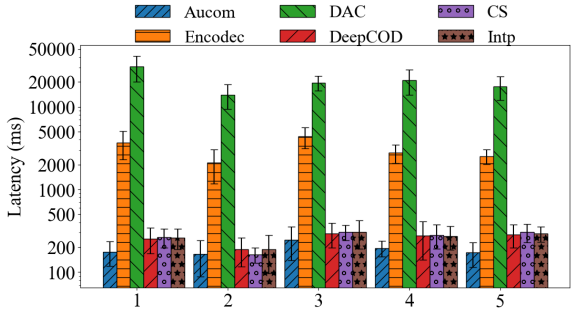


Fig. 14: Average encoding latency (ms) across different smartphones and compression methods.

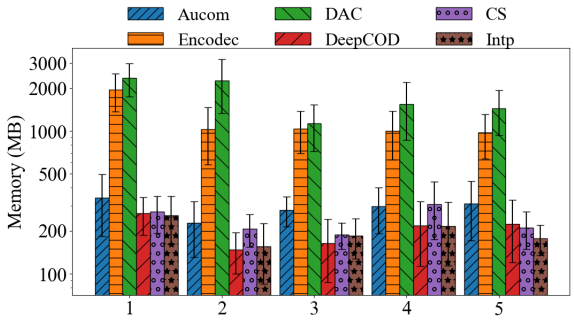


Fig. 15: Memory usage (MB) during 1-minute audio compression on different devices.

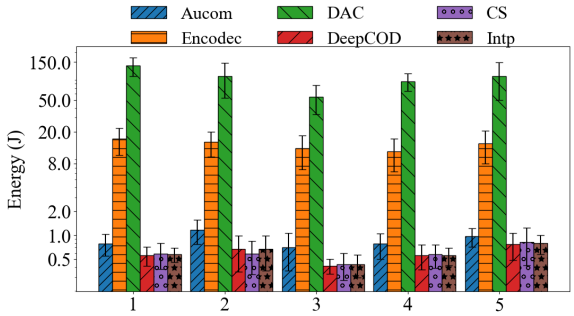


Fig. 16: Comparison of energy consumption per 1-minute audio compression across different smartphone devices.

To reduce unnecessary computation in quiet environments, the speech enhancement module in AUCOM is conditionally activated. Specifically, before each recording session, the system performs a lightweight pre-recording energy check to estimate ambient noise levels. If the background noise exceeds a predefined threshold, the enhancement module is enabled; otherwise, it remains inactive. This adaptive strategy ensures robustness to dynamic acoustic environments while maintaining low resource usage on edge devices.

In practice, the system captures a short segment (0.5–1.0s) of background audio and computes its average root mean square (RMS) energy. A predefined threshold of 0.02 (in normalized amplitude) is used to determine whether the environment is considered noisy. This check introduces negligible overhead—less than 1 ms latency and minimal memory usage—making it suitable for real-time operation on mobile devices.

To evaluate the resource consumption of edge devices under normal conditions, we conducted tests focusing solely on the performance of Mel-frequency feature extraction and data compression. This assessment aims to measure the computational cost associated with fundamental audio processing tasks, providing insights into the baseline resource requirements without the influence of noise reduction operations. For each 1-minute audio signal recorded by the edge device's microphone, signal processing and compression were conducted, and this process was repeated for 20 audio signals.

We first validate the real-time performance of the AUCOM system by deploying its compression and encoding modules on five representative smartphones. As described in Sec. 5.2, all models are tested using the default CPU-based ONNX runtime, without GPU acceleration. For each device, we record the total time required to compress 20 one-minute audio clips and compute the average latency, memory usage, and energy consumption.

To further contextualize the efficiency of AUCOM, we compare its performance with five representative baseline systems: Encodec [9], DAC [8], DeepCOD [88], CS [89], and Intp. The results are presented in Fig. 16, Fig. 14, and Fig. 15. Compared to the neural network-based method DAC and Encodec, AUCOM demonstrates superior performance across all dimensions. Specifically:

- **Latency:** AUCOM achieves an average encoding latency of $\sim 0.2s$ (200ms) per clip, whereas neural codec baselines (e.g., DAC, Encodec) exhibit latency in the range of 2–30 seconds (Fig. 14).
- **Memory usage:** AUCOM requires less than 400MB of RAM during runtime on all devices, whereas DAC and Encodec demand over 1GB and up to 2.5GB, which may hinder background deployment on resource-constrained devices (Fig. 15).
- **Energy consumption:** AUCOM consistently consumes less than 1.2J per 1-minute audio clip across all devices, significantly lower than Encodec ($\sim$ 10–20J) and DAC ($\sim$ 50–150J), as shown in Fig. 16.

These results confirm that AUCOM fully satisfies real-time requirements on mainstream smartphones. For example, on the Xiaomi 11 Pro, the system consumes only 0.79J to compress a 1-minute audio file. Given a 5000mAh battery and a nominal voltage of 3.8V, this implies continuous audio compression can be sustained for approximately $h = \frac{5000mAh \times 3.8V \times 3.6}{0.79J \times 60} \approx 1443$ hours without recharging. This efficiency highlights the suitability of AUCOM for always-on, daily mobile usage.

Smartphones	CPU↑	GPU↑	MEM↑	UX↑	AI↑	Chipset Model	Memory Size (RAM)	Android OS version
Xiaomi 11 Pro	210273	238996	170166	193928	496976	Qualcomm Snapdragon 888	8GB	14
vivo X80	368452	361166	222312	259556	111173	MediaTek Dimensity 9000	8GB	12
Huawei nova 12	183981	139625	133765	136185	86747	Qualcomm Snapdragon 778G	8GB	12 (Harmony OS 4.2.0)
Google Pixel 6 pro	183604	174109	148578	169700	340971	Google Tensor	12GB	14
Honor X40 GT	192461	216011	144437	184854	509907	Qualcomm Snapdragon 888	12 GB	14

TABLE 4: Comparison of computation micro benchmark tests and detailed device information on test smartphones. Huawei nova 12 is equipped with HarmonyOS 4.2.0, a customized operating system that remains compatible with Android 12.

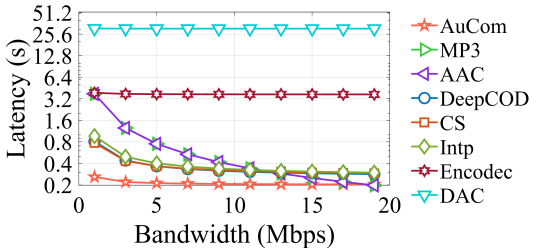


Fig. 17: Comparative analysis of transmission latency of different compression systems.

5.5.3 *The impact of varying bandwidths and transmission protocols on system performance.*

Furthermore, we evaluate the system latency under varying upload bandwidths. We test the Xiaomi 11 Pro with upload bandwidths ranging from 1 to 20 Mbps, using AUCOM, MP3, AAC, DeepCoD, CS, Intp, Encodec and DAC to process and transmit 20 sets of 1-minute audio, measuring the average latency for each case.

The results in Fig. 17 demonstrate that under limited upload bandwidth conditions—such as network congestion due to multiple concurrent devices or weak network signals caused by distance from the coverage center—AUCOM's latency is significantly lower than that of direct MP3 transmission and all other benchmark methods. Specifically, when the available bandwidth per device is limited to 1Mbps, the system latency is reduced by 93.10% compared to MP3. Additionally, AUCOM is suitable for transmission protocols in bandwidth-constrained IoT devices, such as ZigBee (with a bandwidth of 20-250Kbps) and long-range LoRa (with a bandwidth of approximately 10Kbps). AUCOM utilizes only 1Kbps of bandwidth to achieve general audio compression, making it capable of meeting real-time transmission demands under various bandwidth limitations.

6 LIMITATION & FUTURE WORK

Real-Time Processing Constraints. Although AUCOM enables low-latency audio compression, real-time denoising and Mel-spectrogram processing still pose challenges for low-power edge devices. Future work will explore model compression techniques—such as pruning, quantization, and distillation—as well as efficient architectures (e.g., depth-wise convolutions, lightweight Transformers) to reduce latency and energy consumption while maintaining performance.

Privacy and Security Considerations. While AUCOM reduces raw audio exposure through Mel-spectrogram compression, residual speech patterns may still pose privacy risks. Future work will explore privacy-preserving techniques—such as homomorphic encryption, secure multi-party computation, and differential privacy—to ensure secure edge-to-server transmission without sacrificing efficiency.

Expanding Dataset Diversity. While our evaluation on VoiceBank+DEMAND aligns with common practice, its limited speaker diversity may constrain generalization. In future work, we will explore larger and more varied datasets with broader speaker, accent, and noise coverage to better assess AUCOM's robustness in real-world conditions.

Exploring Alternative Efficient Attention Mechanisms. We adopt Performer for linear-time efficiency, while noting alternatives (Longformer [97], Squeezeformer [98]) offer different accuracy/efficiency trade-offs. Future work will benchmark these within our framework to guide lightweight attention for edge deployment.

Extending to hierarchical edge-to-server architectures. Our framework is modular and compatible with both two-layer (edge device–cloud) and three-layer (edge device–edge server–cloud) deployments. In a two-layer architecture, offloading decisions are relatively straightforward but highly sensitive to end-to-end bandwidth and latency fluctuations, since the device communicates directly with the cloud. In contrast, a three-layer architecture introduces an intermediate edge server, which provides additional flexibility for task placement (e.g., preprocessing or partial inference at the edge), but also increases the complexity of scheduling and coordination. Offloading decisions (e.g., decoder/inference placement) are inherently dynamic and must adapt to real-time network conditions, device capabilities, and server load. Since researches [10], [11] analyze the relative difficulty and trade-offs of offloading in two- versus three-layer pipelines. We plan to integrate and empirically evaluate these mechanisms in future work.

Lightweight adaptivity of noise gating. Beyond the fixed frame-energy threshold (0.02), we plan to add a lightweight adaptive gate that updates the threshold online from recent signal statistics with negligible compute or power cost. This thin wrapper only controls the enhancement branch, using short-memory stats with basic hysteresis to stabilize across quiet or noisy scenes. It improves robustness without altering the architecture or energy and latency budgets.

7 CONCLUSION

In this paper, we propose an efficient cloud-edge audio streaming architecture based on Mel spectrum. The system deploys Mel feature extraction and quantization models on the edge, effectively breaking through the bottleneck of traditional audio compression and achieving extreme compression ratio and low latency performance. The system builds decompression modules to support high-precision processing of tasks such as speech recognition, speech emotion recognition, and audio classification in the cloud. This architecture provides strong technical support for real-time audio stream processing and shows wide application potential.

ACKNOWLEDGMENT

This work is supported in part by National Natural Science Foundation of China (No. 61936015, No.6240071246), Natural Science Foundation of Shanghai (No. 24ZR1430600) and Shanghai Key Laboratory of Trusted Data Circulation and Governance, and Web3.

REFERENCES

[1] J. Yuan, "The competition and comparison among travel service giants–a case study of didi and uber," in *Global Dialogue on Media Dynamics, Trends and Perspectives on Public Relations and Communication*. CRC Press, 2025, pp. 648–653.

[2] M. van Beurden and A. Weaver, "Rfc 9639: Free lossless audio codec (flac)," 2024.

[3] R. Ahmed, M. S. Islam, and J. Uddin, "Optimizing apple lossless audio codec algorithm using nvidia cuda architecture," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 8, no. 1, pp. 70–75, 2018.

[4] B. De Man and J. D. Reiss, "Ape: Audio perceptual evaluation toolbox for matlab," in *Audio Engineering Society Convention 136*. Audio Engineering Society, 2014.

[5] S. Whibley, M. Day, P. May, and M. Pennock, "Wav format preservation assessment," *British Library: London, UK*, 2016.

[6] J. Sterne, *MP3: The meaning of a format*. Duke University Press, 2020.

[7] K. Brandenburg, "Mp3 and aac explained," in *Audio Engineering Society Conference: 17th International Conference: High-Quality Audio Coding*. Audio Engineering Society, 1999.

[8] R. Kumar *et al.*, "High-fidelity audio compression with improved rvqgan," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[9] A. Défossez *et al.*, "High fidelity neural audio compression," *arXiv preprint arXiv:2210.13438*, 2022.

[10] W. Zhang *et al.*, "Elf: accelerate high-resolution mobile deep vision with content-aware parallel offloading," in *MobiCom*, 2021.

[11] G. Sun, Z. Wang, H. Su, H. Yu, B. Lei, and M. Guizani, "Profit maximization of independent task offloading in mec-enabled 5g internet of vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 11, pp. 16 449–16 461, 2024.

[12] A. Vaswani *et al.*, "Attention is all you need," *NeurIPS*, vol. 30, 2017.

[13] Q. Zhang *et al.*, "Transformer transducer: A streamable speech recognition model with transformer encoders and rnn-t loss," in *IEEE ICASSP*. IEEE, 2020, pp. 7829–7833.

[14] B. Schuller *et al.*, "Hidden markov model-based speech emotion recognition," in *ICASSP*, vol. 2. IEEE, 2003, pp. II–1.

[15] G. Trigeorgis *et al.*, "Adieu features? end-to-end speech emotion recognition using a deep convolutional recurrent network," in *ICASSP*. IEEE, 2016, pp. 5200–5204.

[16] A. Stuhlsatz *et al.*, "Deep neural networks for acoustic emotion recognition: Raising the benchmarks," in *ICASSP*, 2011.

[17] S. Hershey *et al.*, "Cnn architectures for large-scale audio classification," in *IEEE ICASSP*. IEEE, 2017, pp. 131–135.

[18] K. Palanisamy, D. Singhania, and A. Yao, "Rethinking cnn models for audio classification," *arXiv preprint arXiv:2007.11154*.

[19] X. Liu *et al.*, "Cat: Causal audio transformer for audio classification," in *IEEE ICASSP*. IEEE, 2023, pp. 1–5.

[20] A. Radford *et al.*, "Robust speech recognition via large-scale weak supervision," in *ICML*. PMLR, 2023, pp. 28 492–28 518.

[21] J.-B. Delbrouck *et al.*, "A transformer-based joint-encoding for emotion recognition and sentiment analysis," *ACL 2020*, p. 1, 2020.

[22] Y. Gong *et al.*, "Psla: Improving audio tagging with pretraining, sampling, labeling, and aggregation," *IEEE/ACM TASLP*, 2021.

[23] H. Xu *et al.*, "Learning alignment for multimodal emotion recognition from speech," *Proc. Interspeech 2019*, pp. 3569–3573, 2019.

[24] Y. Zhang *et al.*, "Towards end-to-end speech recognition with deep convolutional neural networks," *arXiv preprint arXiv:1701.02720*, 2017.

[25] A. G. Howard *et al.*, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*.

[26] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *IEEE CVPR*, 2018, pp. 7132–7141.

[27] S. Karita *et al.*, "A comparative study on transformer vs rnn in speech applications," in *2019 IEEE ASRU workshop*, 2019.

[28] W. Zhu and M. Omar, "Multiscale audio spectrogram transformer for efficient audio classification," in *IEEE ICASSP*, 2023.

[29] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE TASSP*, vol. 28, no. 4, pp. 357–366, 1980.

[30] K. Ito and L. Johnson, "The lj speech dataset," https://keithito.com/LJSpeechDataset/, 2017.

[31] V. Panayotov *et al.*, "Librispeech: an asr corpus based on public domain audio books," in *IEEE ICASSP*, 2015.

[32] C. Busso *et al.*, "Iemocap: Interactive emotional dyadic motion capture database," *Language resources and evaluation*, vol. 42, 2008.

[33] J. Tang, L. Zhang, Y. Lu, D. Ding, L. Yang, Y. Chen, M. Bian, X. Li, and G. Xue, "Vcemo: Multi-modal emotion recognition for chinese voiceprints," *arXiv preprint arXiv:2408.13019*, 2024.

[34] D. He, Z. Yang, W. Peng, R. Ma, H. Qin, and Y. Wang, "Elic: Efficient learned image compression with unevenly grouped space-channel contextual adaptive coding," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5718–5727.

[35] J. Li, B. Li, and Y. Lu, "Deep contextual video compression," *Advances in Neural Information Processing Systems*, vol. 34, pp. 18 114–18 125, 2021.

[36] F. Mentzer, G. Toderici, D. Minnen, S.-J. Hwang, S. Caelles, M. Lucic, and E. Agustsson, "Vct: A video compression transformer," *arXiv preprint arXiv:2206.07307*, 2022.

[37] J. Ballé *et al.*, "Density modeling of images using a generalized normalization transformation," *arXiv preprint arXiv:1511.06281*, 2015.

[38] L. Theis *et al.*, "Lossy image compression with compressive autoencoders," in *ICML*, 2022.

[39] O. Rippel and L. Bourdev, "Real-time adaptive image compression," in *ICML*, 2017, pp. 2922–2930.

[40] J. Duda, "Asymmetric numeral systems," *arXiv preprint arXiv:0902.0271*, 2009.

[41] J. S. Vitter, "Design and analysis of dynamic huffman codes," *Journal of the ACM (JACM)*, vol. 34, no. 4, pp. 825–845, 1987.

[42] J. B. Connell, "A huffman-shannon-fano code," *Proceedings of the IEEE*, vol. 61, no. 7, pp. 1046–1047, 1973.

[43] Y. Wei *et al.*, "Adastreamlite: Environment-adaptive streaming speech recognition on mobile devices," *PACM IMWUT*, vol. 7, no. 4, pp. 1–29, 2024.

[44] D. Duan, Y. Chen, W. Xu, and T. Li, "Earse: Bringing robust speech enhancement to cots headphones," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 7, no. 4, Jan. 2024. [Online]. Available: https://doi.org/10.1145/3631447

[45] S. Pascual, A. Bonafonte, and J. Serra, "Segan: Speech enhancement generative adversarial network," *arXiv preprint arXiv:1703.09452*, 2017.

[46] A. Pandey and D. Wang, "Tcnn: Temporal convolutional neural network for real-time speech enhancement in the time domain," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6875–6879.

[47] A. Defossez, G. Synnaeve, and Y. Adi, "Real time speech enhancement in the waveform domain," *arXiv preprint arXiv:2006.12847*, 2020.

[48] E. Kim and H. Seo, "Se-conformer: Time-domain speech enhancement using conformer." in *Interspeech*, 2021, pp. 2736–2740.

[49] Z. Kong, W. Ping, A. Dantrey, and B. Catanzaro, "Speech denoising in the waveform domain with self-attention," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 7867–7871.

[50] C. Valentini-Botinhao and J. Yamagishi, "Speech enhancement of noisy and reverberant speech for text-to-speech," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 8, pp. 1420–1433, 2018.

[51] Y. Ai, J.-X. Zhang, L. Chen, and Z.-H. Ling, "Dnn-based spectral enhancement for neural waveform generators with low-bit quantization," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 7025–7029.

[52] Y. Xu, J. Du, L.-R. Dai, and C.-H. Lee, "A regression approach to speech enhancement based on deep neural networks," *IEEE/ACM transactions on audio, speech, and language processing*, vol. 23, no. 1, pp. 7–19, 2014.

[53] J. Kim, M. El-Khamy, and J. Lee, "T-gsa: Transformer with gaussian-weighted self-attention for speech enhancement," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6649–6653.

[54] Y. Hu, Y. Liu, S. Lv, M. Xing, S. Zhang, Y. Fu, J. Wu, B. Zhang, and L. Xie, "Dccrn: Deep complex convolution recurrent network for phase-aware speech enhancement," *arXiv preprint arXiv:2008.00264*, 2020.

[55] S. Zhao, B. Ma, K. N. Watcharasupat, and W.-S. Gan, "Frcrn: Boosting feature representation using frequency recurrence for monaural speech enhancement," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 9281–9285.

[56] S. Zhao and B. Ma, "Mossformer: Pushing the performance limit of monaural speech separation using gated single-head transformer with convolution-augmented joint self-attentions," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023, pp. 1–5.

[57] S. Zhao, Y. Ma, C. Ni, C. Zhang, H. Wang, T. H. Nguyen, K. Zhou, J. Q. Yip, D. Ng, and B. Ma, "Mossformer2: Combining transformer and rnn-free recurrent network for enhanced time-domain monaural speech separation," in *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024, pp. 10 356–10 360.

[58] K. Paliwal, K. Wójcicki, and B. Shannon, "The importance of phase in speech enhancement," *speech communication*, vol. 53, no. 4, pp. 465–494, 2011.

[59] R. Cao, S. Abdulatif, and B. Yang, "Cmgan: Conformer-based metric gan for speech enhancement," *arXiv preprint arXiv:2203.15149*, 2022.

[60] D. S. Williamson, Y. Wang, and D. Wang, "Complex ratio masking for monaural speech separation," *IEEE/ACM transactions on audio, speech, and language processing*, vol. 24, no. 3, pp. 483–492, 2015.

[61] Z.-Q. Wang, G. Wichern, and J. Le Roux, "On the compensation between magnitude and phase in speech separation," *IEEE Signal Processing Letters*, vol. 28, pp. 2018–2022, 2021.

[62] D. Yin, C. Luo, Z. Xiong, and W. Zeng, "Phasen: A phase-and-harmonics-aware speech enhancement network," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, 2020, pp. 9458–9465.

[63] Y.-X. Lu, Y. Ai, and Z.-H. Ling, "Mp-senet: A speech enhancement model with parallel denoising of magnitude and phase spectra," *arXiv preprint arXiv:2305.13686*, 2023.

[64] K. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlos, P. Hawkins, J. Davis, A. Mohiuddin, L. Kaiser *et al.*, "Rethinking attention with performers," *arXiv preprint arXiv:2009.14794*, 2020.

[65] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[66] D. Ulyanov, "Instance normalization: The missing ingredient for fast stylization," *arXiv preprint arXiv:1607.08022*, 2016.

[67] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[68] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[69] S. E. Finder, R. Amoyal, E. Treister, and O. Freifeld, "Wavelet convolutions for large receptive fields," in *European Conference on Computer Vision*. Springer, 2024, pp. 363–380.

[70] S.-W. Fu, C. Yu, T.-A. Hsieh, P. Plantinga, M. Ravanelli, X. Lu, and Y. Tsao, "Metricgan+: An improved version of metricgan for speech enhancement," *arXiv preprint arXiv:2104.03538*, 2021.

[71] A. W. Rix, J. G. Beerends, M. P. Hollier, and A. P. Hekstra, "Perceptual evaluation of speech quality (pesq)-a new method for speech quality assessment of telephone networks and codecs," in *2001 IEEE international conference on acoustics, speech, and signal processing. Proceedings (Cat. No. 01CH37221)*, vol. 2. IEEE, 2001, pp. 749–752.

[72] Y. Ai and Z.-H. Ling, "Neural speech phase prediction based on parallel estimation architecture and anti-wrapping losses," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023, pp. 1–5.

[73] V. K. Goyal, "Theoretical foundations of transform coding," *IEEE Signal Processing Magazine*, vol. 18, no. 5, pp. 9–21, 2001.

[74] J. Rissanen and G. Langdon, "Universal modeling and coding," *IEEE Transactions on Information Theory*, vol. 27, no. 1, pp. 12–23, 1981.

[75] A. Graves and A. Graves, "Long short-term memory," *Supervised sequence labelling with recurrent neural networks*, pp. 37–45, 2012.

[76] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression," *arXiv preprint arXiv:1611.01704*, 2016.

[77] G. Toderici *et al.*, "Full resolution image compression with recurrent neural networks," in *IEEE CVPR*, 2017.

[78] C. Valentini-Botinhao, X. Wang, S. Takaki, and J. Yamagishi, "Investigating rnn-based speech enhancement methods for noise-robust text-to-speech." in *SSW*, 2016, pp. 146–152.

[79] J. F. Gemmeke *et al.*, "Audio set: An ontology and human-labeled dataset for audio events," in *IEEE ICASSP*. IEEE, 2017, pp. 776–780.

[80] K. J. Piczak, "ESC: Dataset for Environmental Sound Classification," in *ACM Multimedia*. ACM Press, pp. 1015–1018. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2733373.2806390

[81] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *arXiv preprint arXiv:1804.03209*, 2018.

[82] I. Loshchilov, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[83] D. Kingma, "Adam: a method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[84] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *NeurIPS*, vol. 32, 2019.

[85] S. Imambi, K. B. Prakash, and G. Kanagachidambaresan, "Pytorch," *Programming with TensorFlow: solution for edge computing applications*, pp. 87–104, 2021.

[86] P. Jajal, W. Jiang, A. Tewari, E. Kocinare, J. Woo, A. Sarraf, Y.-H. Lu, G. K. Thiruvathukal, and J. C. Davis, "Interoperability in deep learning: A user survey and failure analysis of onnx model converters," in *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2024, pp. 1466–1478.

[87] T. Hagos, "Android studio," in *Learn Android Studio 3: Efficient Android App Development*. Springer, 2018, pp. 5–17.

[88] S. Yao, J. Li, D. Liu, T. Wang, S. Liu, H. Shao, and T. Abdelzaher, "Deep compressive offloading: Speeding up neural network inference by trading edge computation for network latency," in *Proceedings of the 18th conference on embedded networked sensor systems*, 2020, pp. 476–488.

[89] D. You, J. Xie, and J. Zhang, "Ista-net++: Flexible deep unfolding network for compressive sensing," in *2021 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2021, pp. 1–6.

[90] S.-W. Fu, C.-F. Liao, Y. Tsao, and S.-D. Lin, "Metricgan: Generative adversarial networks based black-box metric scores optimization for speech enhancement," in *International Conference on Machine Learning*. PmLR, 2019, pp. 2031–2041.

[91] F. Dang, H. Chen, and P. Zhang, "Dpt-fsnet: Dual-path transformer based full-band and sub-band fusion network for speech enhancement," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 6857–6861.

[92] D. Yin, Z. Zhao, C. Tang, Z. Xiong, and C. Luo, "Tridentse: Guiding speech enhancement with 32 global tokens," *arXiv preprint arXiv:2210.12995*, 2022.

[93] G. Yu, A. Li, C. Zheng, Y. Guo, Y. Wang, and H. Wang, "Dual-branch attention-in-attention transformer for single-channel speech enhancement," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 7847–7851.

[94] J. Rissanen and G. G. Langdon, "Arithmetic coding," *IBM Journal of research and development*, vol. 23, no. 2, pp. 149–162, 1979.

[95] B. Rosen, J. Goodwin, and J. Vidal, "Adaptive range coding," *Advances in neural information processing systems*, vol. 3, 1990.

[96] Antutu, 2023. [Online]. Available: https://www.antutu.com/en/doc/index.htm

[97] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," *arXiv preprint arXiv:2004.05150*, 2020.

[98] S. Kim, A. Gholami, A. Shaw, N. Lee, K. Mangalam, J. Malik, M. W. Mahoney, and K. Keutzer, "Squeezeformer: An efficient transformer for automatic speech recognition," *Advances in Neural Information Processing Systems*, vol. 35, pp. 9361–9373, 2022.