TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

A

PROJECT REPORT

ON

NEURAL AUDIO CODEC

SUBMITTED BY:

SUBODH BARAL (PUL075BCT088)

TAPENDRA PANDEY (PUL075BCT093)

ACHYUT BURLAKOTI (PUL075BCT098)

SIJAL BARAL (PUL075BCT100)

SUBMITTED TO:

DEPARTMENT OF ELECTRONICS & COMPUTER ENGINEERING

Apr 29, 2023

Page of Approval

TRIBHUVAN UNIVERSIY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS
DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

The undersigned certifies that they have read and recommended to the Institute of Engineering for acceptance of a project report entitled **"Neural Audio Codec"** submitted by **Subodh Baral**, **Tapendra Pandey**, **Achyut Burlakoti**, **Sijal Baral** in partial fulfillment of the requirements for the Bachelor's degree in Electronics & Computer Engineering.

....

Supervisor

Dr. Sashidhar Ram Joshi, Phd.

Professor

Department of Electronics and Computer Engineering,

Pulchowk Campus, IOE, TU.

....

Internal examiner

Person B

Assistant Professor

Department of Electronics and Computer Engineering,

Pulchowk Campus, IOE, TU.

....

External examiner

Person C

Assistant Professor

Department of Electronics and Computer Engineering,

Pulchowk Campus, IOE, TU.

Date of approval:

Copyright

Acknowledgments

Abstract

Neural audio codecs that use end-to-end approaches have gained popularity due to their ability to learn efficient audio representations through data-driven methods, without relying on handcrafted signal processing components. This research paper evaluates the performance of Neural Audio Codec in comparison to traditional audio codecs Opus and EVS in terms of audio quality and efficiency. The study highlights the limitations of existing audio codecs in leveraging the abundant data available in the audio compression pipeline and proposes deep learning-based models as a potential solution. The paper reviews recent advancements in deep learning-based audio synthesis and representation learning and explores the potential of deep learning-based audio codecs in enhancing compression efficiency. The study also addresses the limitations of existing models, including slower training times and increased memory requirements, by releasing open-source code and pre-trained models for further research and improvement. Experimental results show that our approach has comparable performance to widely used commercial codec OPUS at low bitrate, and a slight drop in performance compared to current deep learning-based frameworks but at the expense of significant improvement in speed and memory requirements. We have released our code and pre-trained models at https://github.com/AchyutBurlakoti/Neural-Audio-Compression for further research and improvement.

Keywords: Audio Compression, Deep Learning, Audio Codec, Pre-trained models

Contents

List of Figures

List of Tables

List of Abbreviations

BPS	Bits Per Second
CNN	Convolutional Neural Networks
DCT	Discrete Cosine Transform
DTMF	Dual Tone Multi-Frequency
EVS	Enhanced Voice Services
GAN	Generative Adversarial Networks
MUSHRA	Multiple Stimuli with Hidden Reference and Anchor
PESQ	Perceptual Evaluation of Speech Quality
POLQA	Perceptual Objective Listening Quality Assessment
VAEs	Variational Autoencoders
VQ	Vector Quantizer

1. Introduction

1.1   Background

According to recent research, traditional audio codecs such as Opus and EVS are considered state-of-the-art codecs, as they employ different coding tools, including Linear Predictive Coding (LPC), Code-Excited Linear Prediction (CELP), and Modified Discrete Cosine Transform (MDCT), to achieve high coding efficiency, low latency, and compatibility with various content types, bitrates, and sampling rates [3] [2]. In this study, a subjective evaluation to compare Neural Audio Codec with Opus and EVS in terms of audio quality and performance was conducted. Current audio standards have limitations in leveraging the abundant data available in the audio compression pipeline, which can potentially enhance the prediction process for more efficient entropy coding. Existing conventional approaches [3, 2] employ block-based structures and rely on handcrafted modules [5, 28, 14]. Although neural networks have been utilized to achieve localized improvements in audio codecs, an end-to-end model could further enhance the overall compression efficiency. Nevertheless, deep learning-based models for audio coding are still in their early stages [38].

Recent advancements in deep learning-based audio synthesis and representation learning [4, 32, 22, 11, 18, 33] have brought a significant breakthrough in audio compression. Inspired by this, several deep learning-based audio codecs have been proposed [39, 37, 6]. These approaches exploit the powerful representation ability of neural networks that learn the optimal transformation of audio waveforms to a lower-dimensional latent space. The model is composed of two parts: the encoder, which learns the prior distribution by transforming the waveform into the latent space, and the decoder, which learns the posterior distribution by predicting the waveform from the latent variable. This enables the model to learn the distribution of audio data using both the prior and posterior probabilities, expressed as $p(x, z) = \sum_z p(x|z)p(z)$. Despite the improvement in compression and quantization of audio signals, these models have several limitations, including slower training times and increased memory requirements. Additionally, some of these models have closed-source code, making it difficult for others to explore the pipeline further. In this work, we aim to address these limitations by releasing our open-source code and pre-trained models at https://github.com/AchyutBurlakoti/Audio-Compression for further research and improvement.

1

1.2   Objectives

1. Build a modern audio codec, with a neural network approach, which can compress the audio file.

1.3   Scope of the Project

There are many existing audio compression algorithms which are performing well such as MP3, AAC, and Opus, but we believe that they can be optimized further by using state-of-the-art machine learning algorithms.

The current amount of audio information being transmitted over the internet shows how much dependent the world has become on the audio contents be it in the field of communication, entertainment, music production, podcasting and so on. A slight improvement in the existing audio compression algorithms can save a lot of bandwidth which is soon going to be scarce given the huge rate of inflow of internet users worldwide. By using machine learning algorithms, we can potentially achieve higher compression ratios without compromising the audio quality, which can result in significant bandwidth savings and improved user experience.

2.   Theoretical Background
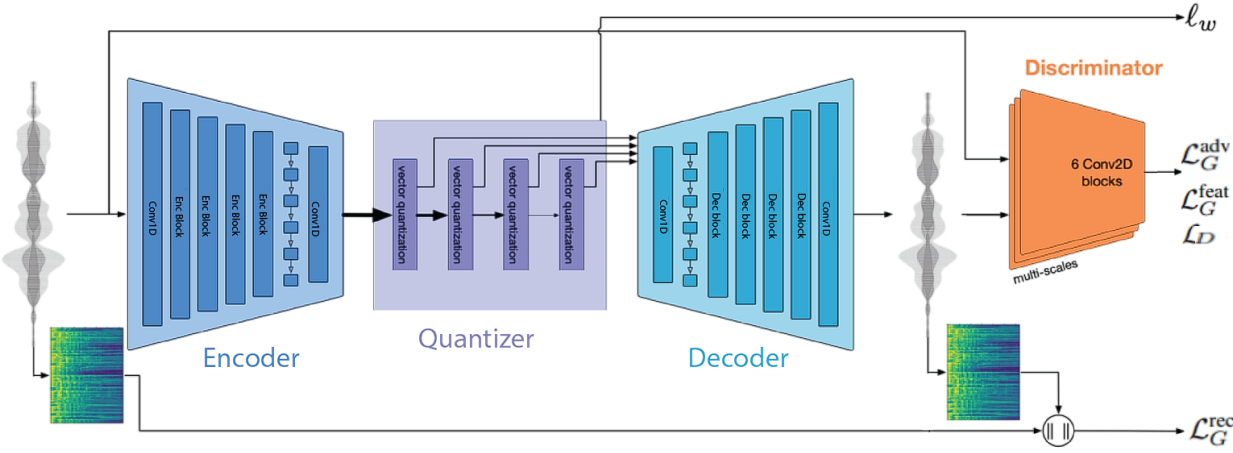
Figure 2.1: Encoder and Decoder Model Architecture

2.1   Overview of Audio Compression

Audio compression techniques aim to reduce the amount of data needed to represent an audio signal while maintaining its perceived quality. There are two main types: lossless and lossy. Lossless compression algorithms use redundancies to achieve compression without any loss of information, while lossy algorithms discard less perceptually important information. Popular lossy formats include MP3, AAC, and Ogg Vorbis, which use psychoacoustic modeling and transform coding. Lossless formats such as FLAC and ALAC exploit the statistical properties of the audio signal without discarding any information.

Encoder

An encoder is an architecture that transforms input data into a compressed, lower-dimensional representation called a "code" or "embedding." The encoder typically consists of multiple layers of neurons that perform non-linear transformations on the input data. These transformations progressively reduce the dimensionality of the input data, resulting in a compressed representation.

Quantization

Quantization is a fundamental process in data compression, and its main job is to discretize a continuous latent space by preparing a codebook. In audio and image compression, quantization is commonly used to represent high-dimensional data with lower-dimensional embeddings. The quantizer prepares the codebook for these embeddings, allowing us to store the index of their nearest neighbor in the codebook. This process is called vector quantization, and it involves grouping similar embeddings together into clusters. The codebook consists of the centroid of each cluster, which is represented by a discrete symbol.

Decoder

Decoder is multi-layered neural architecture that takes a compressed, low-dimensional representation of the input data generated by an encoder and transforms it back into a high-dimensional output representation that can be used for tasks such as reconstruction, classification, or generation.

Discriminator

In the context of audio synthesis, a discriminator is a neural network that is trained to distinguish between real and synthetic audio signals. The discriminator is a key component of generative adversarial networks (GANs), a type of deep learning model that is widely used for audio synthesis. The discriminator is trained to learn the characteristics of real audio signals and differentiate them from synthesized signals generated by the generator network. By doing so, it provides feedback to the generator network on how to improve the quality of the synthesized audio.

2.2   Vector Quantization

The signal processing method known as vector quantization (VQ) permits the modeling of probability density functions through the distribution of prototype vectors. Its initial application was for data compression. It divides a big collection of points (vectors) into groups that each include roughly the same number of points that are nearest to them.

2.3   DCT

The Discrete Cosine Transform (DCT) is a transformation method commonly used in signal processing and data compression. It represents a set of finite data points as a combination of cosine functions oscillating at various frequencies. The utilization of cosine functions instead of sine functions is crucial for compression because it requires fewer cosine functions to approximate a typical signal. However, for differential equations, the use of cosines indi-

cates a specific selection of boundary conditions. For two dimensional signals, DCT can be represented mathematically as:

$$X_{u,v} = \frac{2}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} x_{x,y} \cos\left[\frac{\pi}{N}\left(x + \frac{1}{2}\right)u\right] \cos\left[\frac{\pi}{N}\left(y + \frac{1}{2}\right)v\right]$$

where $0 \leq u, v < N$ are the frequency indices, and $x_{x,y}$ is the pixel value of the input signal at location (x, y).

2.4   CNN

A Convolutional Neural Network (CNN) consists of an input and an output layer, as well as several hidden layers. The hidden layers in a CNN usually comprise a sequence of convolutional layers that perform dot product or multiplication. The activation function employed is typically a RELU layer, followed by further convolutional layers such as pooling, fully connected and normalization layers, collectively referred to as hidden layers because their inputs and outputs are masked by the activation function and final convolution.

Although these layers are commonly called convolutions, this is only a convention. In reality, mathematically, they perform a sliding dot product or cross-correlation. This has an impact on the matrix indices, influencing how the weight is calculated at a particular index point.
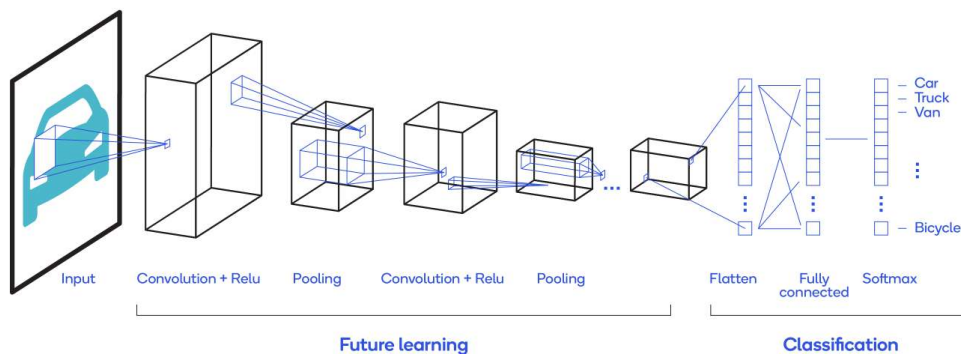
Figure 2.2: CNN architecture.

2.5   Auto-Encoder

An autoencoder is an artificial neural network that operates in an unsupervised manner, and it is designed to learn how to compress and encode data efficiently. Once it has accomplished

this, the network then learns how to reconstruct the original data from the compressed encoded representation, such that the final representation is as close to the original input as possible. Autoencoders are capable of transforming high-dimensional data into a more compact code representation, which can be reconstructed back to the original version with minimal or no differences. The autoencoder is made up of four major components: the encoder, the bottleneck, the decoder, and the reconstruction loss.
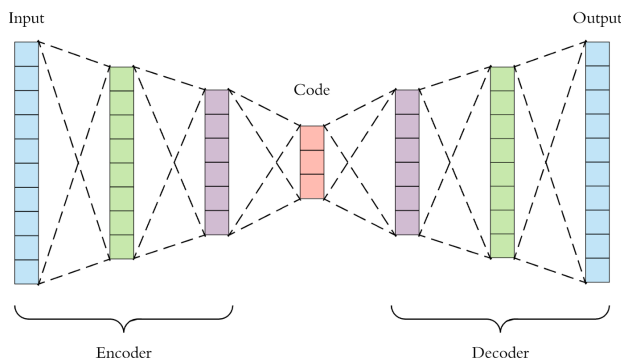
Figure 2.3: Auto Encoder

2.6   Variational AutoEndcoder

A variational autoencoder (VAE), is an artificial neural network architecture introduced by Diederik P. Kingma and Max Welling, belonging to the families of probabilistic graphical models and variational Bayesian methods. Because to its similarity in architecture, variational autoencoders are frequently compared to the autoencoder model, despite having vastly different objectives and mathematical formulations. Statistical inference issues (such determining the value of one random variable from another randomly generated value) can be reformulated as statistical optimization issues using variational autoencoders (i.e find the parameter values that minimize some objective function).
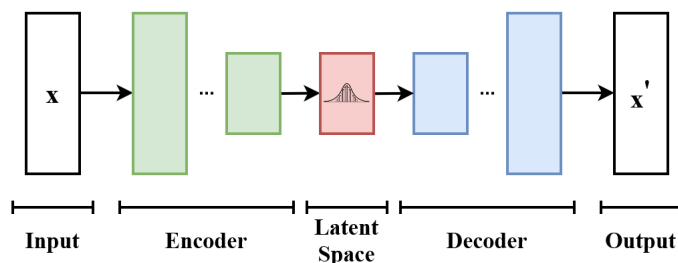
Figure 2.4: Variational AutoEncoder

3.   Literature Review

Traditional Audio Codec

Traditional audio codecs are based on block-based structures and rely on handcrafted modules for audio compression. These codecs employ subband filtering, psychoacoustic modeling, and transform coding techniques, such as the Modified Discrete Cosine Transform (MDCT), to achieve high compression ratios while maintaining perceptual audio quality. Traditional audio codecs are based on block-based structures and rely on handcrafted modules for audio compression. These codecs employ subband filtering, psychoacoustic modeling, and transform coding techniques, such as the Modified Discrete Cosine Transform (MDCT), to achieve high compression ratios while maintaining perceptual audio quality.

In particular all of the audio codec explained based on this scheme. The development of audio codecs such as EVS, Opus, MPEG audio, and MP3 has enabled digital audio compression for various applications. While EVS and Opus offer high-quality compression, MPEG audio provides different levels of compression and quality, and MP3 is widely used for efficient digital audio storage and transmission. However, these traditional codecs have limitations such as limited compression efficiency, restricted quality at low bit rates, and limited support for multichannel audio. Despite these limitations, they remain widely used due to their compatibility and ease of use. [3, 2, 5, 14].

Neural audio compression

Neural audio codecs that use end-to-end approaches have gained popularity due to their ability to learn efficient audio representations through data-driven methods, without relying on handcrafted signal processing components. These codecs utilize autoencoder networks with quantization of hidden features, and have been applied in early works for speech coding [26], as well as in more recent studies, where a deep convolutional network was used for speech compression [19]. While most of these works target speech coding at low bitrates, several studies have demonstrated the efficient compression of audio using neural networks. For instance, a VQ-VAE speech codec was proposed in [11], which operates at a bitrate of 1.6 kbps.

In recent years, several studies have proposed neural-based audio codecs that rely on quantizing the latent space prior to decoding [27, 31, 12, 25, 20, 21, 16, 17]. One approach, used by Valin and Skoglund, involved conditioning an LPCNet vocoder on hand-crafted

features and a uniform quantizer [31]. Meanwhile, Gârbacea et al. conditioned a WaveNet model on discrete units obtained from a VQ-VAE model [12]. Other approaches have included feeding the Opus codec to a WaveNet in order to improve perceptual quality [27], using a vector quantization layer applied over the latent representation in an auto-encoder [16, 17], and using Gumbel-Softmax for representation quantization.

The SoundStream model developed by Zeghidour et al. is a notable example of this type of work. This model uses a fully convolutional encoder-decoder architecture with Residual Vector Quantization layers, and it is optimized using both reconstruction loss and adversarial perceptual losses [37]. Although there are some variations in the specific methods used in each of these studies, the overall trend towards neural-based audio codecs with quantized latent spaces is clear. In most recent paper, Alexandre et al. proposed a method for high-fidelity neural audio compression using a single multiscale spectrogram adversary for efficient training and high-quality sample production, coupled with Transformer models for up to 40% compression rates, all while maintaining real-time performance for entropy coding. [9]

Audio Synthesis

Recent studies have demonstrated that relying solely on reconstruction to generate audio can lead to the presence of noise in the output [10]. To mitigate this issue, researchers have incorporated discriminators to improve the distribution of input data approximated by the generator, resulting in reduced noise in the synthesized audio [8].

In particular, the use of discriminators from generative adversarial models such as Mel-GAN [24] and HiFiGAN [23] has been shown to be effective in generating high-quality audio with lower computational complexity. These models employ multi-scale waveform discriminators to enhance the quality of the synthesized waveform from the decoder [24]. The use of discriminators has thus emerged as a promising approach to generate high-quality audio [23].

4. Methodology
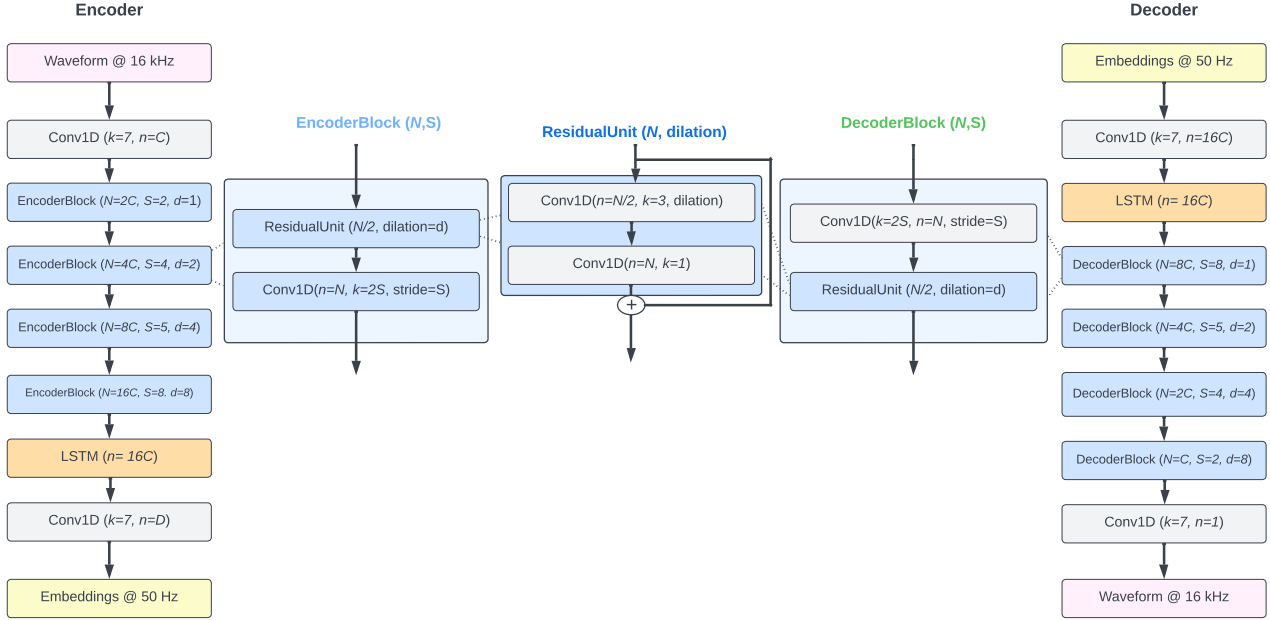
Figure 4.1: Encoder and Decoder Model Architecture

4.1 Encoder Architecture

The encoder structure, as depicted in Figure 4.1, follows the same format as that of the SEANet encoder described in [9, 30]. It comprises a 1D convolution layer with C_{enc} channels, followed by B_{enc} encoder blocks, each with dilated convolutions at varying dilation rates of 1, 2, 4, and 8. The encoder block contains one residual block with a matching dilation, followed by a down-sampling layer in the form of a strided convolution. The number of channels is doubled when down-sampling, starting from C_{enc}. The embeddings are dimensionally reduced to D using a final 1D convolution layer with a 7-length kernel and a stride of 1. All convolutions are causal to ensure real-time inference, with padding applied only to the past in both training and offline inference. No normalization is applied, and the ELU activation function is used [7].

The temporal resampling ratio between the input waveform and the embeddings is determined by the number of encoder blocks, B_{enc}, and the corresponding striding sequence. For example, when B_{enc}=4 and using strides of (2, 4, 5, 8), one embedding is generated for

every $M = 320$ input samples, where M is calculated as (2 . 4 . 5 . 8). Thus, the encoder output, $enc(x)$, is in $\mathbb{R}^{S \times D}$, with $S = T/M$.

This description is based on [3, 2] and explains the encoder architecture employed in the paper.

4.2 Quantizer

The quantizer's objective is to compress the encoder's output, $enc(x)$, to a specific bitrate, R, expressed in bits per second (bps). Joint training of the quantizer, encoder, and decoder is necessary for an end-to-end approach, which can be achieved by backpropagation. The vector quantizer (VQ) proposed for VQ-VAE in [34] satisfies this requirement. It generates a codebook of N vectors to encode each D-dimensional frame of $enc(x)$. Next, $enc(x)$ is mapped to a sequence of one-hot vectors of shape $S \times N$, which requires $\log_2 N$ bits for representation.

Limitation of Vector Quantization

To illustrate, suppose we have a codec with a target bitrate of $R = 3000$ bps. With a striding factor of $M = 320$, one second of audio at a sampling rate of $f_s = 16000$ Hz is represented by S = 50 frames at the encoder output. Each frame is assigned r = 3000 / 50 = 60 bits, given the target bitrate of R. If we were to use a simple vector quantizer, we would need to store a codebook comprising $N = 2^{60}$ vectors, which is not feasible due to its impractical size.

Residual Vector Quantizer

In order to address this issue, we have decided to utilize a Residual Vector Quantizer, which is also known as a multi-stage vector quantizer. This technique involves incorporating N_q layers of VQ in a specific manner. Initially, the unquantized input vector is processed through a first VQ to derive quantization residuals. Subsequently, the residuals are subject to iterative quantization by means of a sequence of additional N_q - 1 vector quantizers, following the guidelines presented in Algorithm 1. The total rate budget is distributed uniformly across all VQs, whereby $r_i = r/N_q = \log_2 N$. If, for instance, $N_q = 6$, each quantizer would make use of a codebook consisting of $N = 2^{r/N_q} = 2^{60/6} = 1024$. By manipulating the N_q parameter, one can balance the computational complexity and coding efficiency to meet a desired target rate budget r.
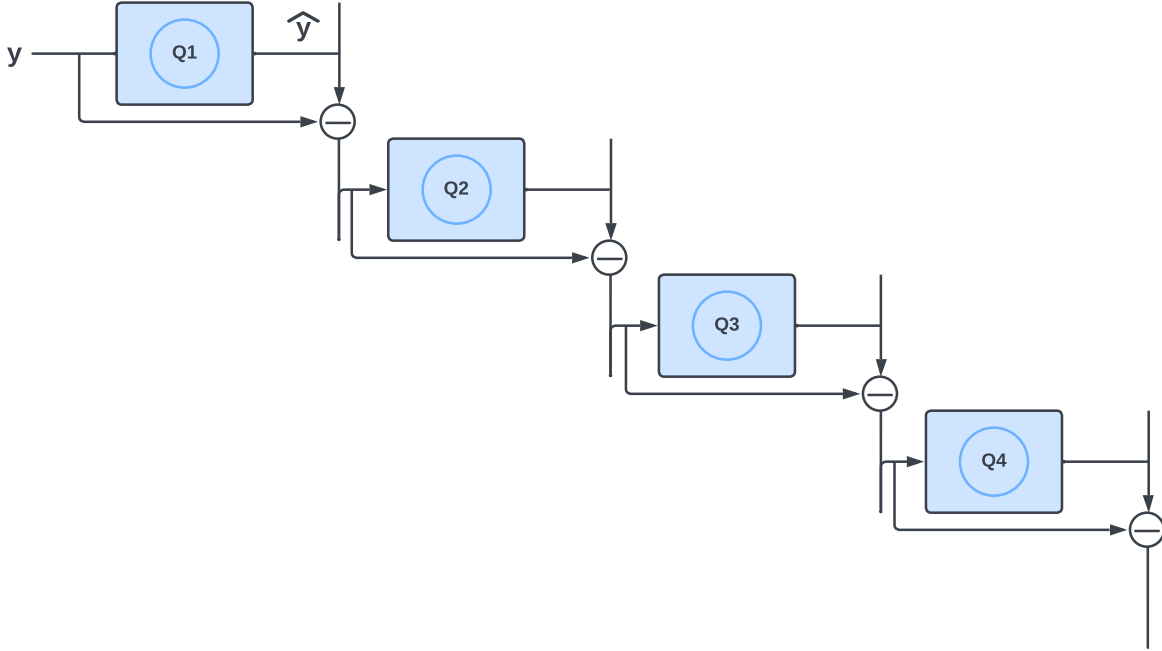
Figure 4.2: Residual Vector Quantizer (for $N_q=4$)

Algorithm 1: Residual Vector Quantization

Input: $y = \mathrm{enc}(x)$ the output of the encoder, vector
   quantizers Q_i for $i = 1..N_q$

Output: the quantized $\hat{y}$

$\hat{y} \leftarrow 0.0$

residual $\leftarrow y$

for $i = 1$ to N_q **do**

   $\hat{y}{+}{=} Q_i(\text{ residual })$

   residual $-{=} Q_i(\text{ residual })$

return $\hat{y}$

Scalable Bitrate

To allow for scalability in bitrate, quantizer dropout is utilized, and residual vector quantization serves as a useful framework for controlling bitrate. The bitrate is determined by the number of VQ layers, N_q, given a fixed codebook size, N, for each layer. In principle, a different model should be trained for each target bitrate, as the vector quantizers are trained jointly with the encoder/decoder. However, it is much more practical to have a single model that can scale to several target bitrates, as it reduces the memory required to store model parameters on both the encoder and decoder side.

To create a model capable of handling multiple bitrates, we make a modification to Algorithm 1. When processing each input example, we randomly sample a value for nq from the range $[1; N_q]$ and only use quantizers Q_i for i = 1 . . . n_q. This is similar to applying structured dropout to the quantization layers. As a result, the model is trained to encode and decode audio at all target bitrates within the range $n_q = 1$ . . . N_q. During the inference phase, we select the value of nq based on the desired bitrate. By implementing this approach, we can create a model that is more memory-efficient than having to train separate models for each target bitrate.

The additive composition of the outputs of each VQ layer progressively refines the quantized embeddings, while keeping the same shape. Hence, no architectural changes are needed in neither the encoder nor the decoder to accommodate different bitrates

4.3   Decoder Architecture

The design of the decoder architecture is similar to that of the encoder architecture, as shown in Figure 3. The decoder block consists of a transposed convolution for up-sampling, followed by residual units, which is identical to the encoder block. We use the reverse order of strides as the encoder to reconstruct a waveform with the same resolution as the input waveform. During up-sampling, the number of channels is halved, and the last decoder block generates C_{dec} channels. A final 1D convolution layer, with a kernel size of 7 and stride 1, with only one filter, is used to project the embeddings back to the waveform domain and produce $\hat{x}$. The same number of channels in both the encoder and decoder is controlled by the same parameter, $C_{enc} = C_{dec} = C$, as illustrated in Figure 4.1.

4.4   Discriminator Architecture

In line with Wang et al. [35], we utilize a multi-scale architecture that includes three discriminators (D1, D2, D3) with the same network structure but differing audio scales. D1 operates on raw audio scale, while D2 and D3 operate on raw audio downsampled by a factor of 2 and 4, respectively. The downsampling is achieved using strided average pooling with a kernel size of 4. The use of multiple discriminators at different scales is justified by the fact that audio has different structures at various levels. This structure is supported by an inductive bias that each discriminator learns features for various frequency ranges of the audio. For instance, the discriminator operating on downsampled audio lacks access to high-frequency components; thus, it is inclined to learn discriminative features based on low-frequency components only.
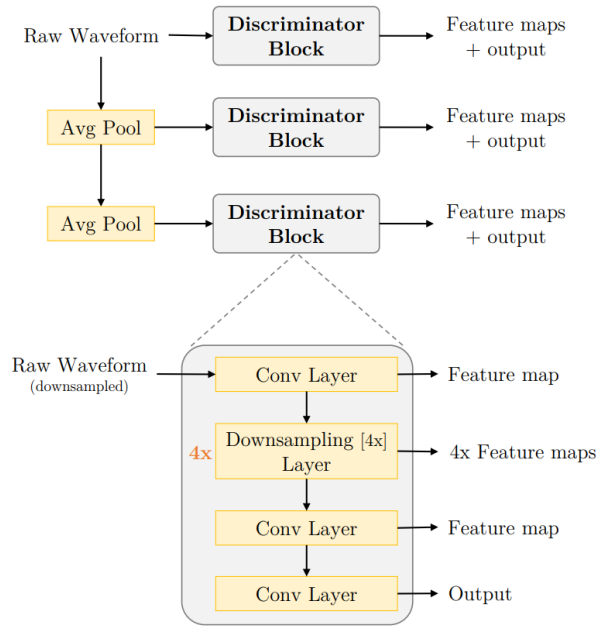
Figure 4.3: Discriminator

4.5 Regularization

Dropout Regularization

Dropout is a popular regularization technique used in deep learning to prevent overfitting of the model. By randomly dropping out some of the neurons in a neural network during training, dropout forces the network to learn redundant representations, which makes it more robust and less likely to overfit. The dropout technique was introduced by Srivastava et al.[29] in their 2014 paper "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". Dropout can be applied to various types of neural network architectures, including fully connected networks, convolutional neural networks (CNNs), and recurrent neural networks (RNNs). To incorporate dropout in a residual block, a dropout layer can be added after the final activation function in the block, before the output is added to the input. The dropout layer can have a dropout rate of p, which determines the fraction of neurons that are randomly dropped out. The value of p should be chosen based on the complexity of the model and the size of the training dataset. In their paper "Identity Mappings in Deep Residual Networks", He et al. [13] showed that using dropout in residual networks can further improve the generalization performance of the model, especially when combined with other regularization techniques. Overall, dropout is a simple and effective regularization technique that can be used to improve the generalization performance of deep learning models, including residual networks.

Time Jitter Regularization

Our goal is for the model to acquire a speech representation that matches the phonetic content that changes gradually within an utterance, which is a mostly steady signal that may suddenly change at the boundaries of phonemes. To avoid co-adaptation of latent vectors, a time-jitter regularizer inspired by dropout [29] was employed in the vq-speech model[6]. During training, a latent vector can replace one or both of its adjacent vectors, similar to dropout, which prevents the model from depending on consistency across groups of tokens. Furthermore, this regularization encourages stability in latent representations over time. A latent vector extracted at a specific time step should also be beneficial at time steps immediately preceding or following it.

4.6 Training Objectives

Reconstruction Loss

In this paper, we adapt a frame-level auxiliary loss between the original and the generated speech as in [36]. The Loss functions that are correlated with perceptual audio quality should be used to train the model.

$$\mathcal{L}_G^{rec} = \lambda_1 \mathcal{L}_{SC}(\boldsymbol{x}, \hat{\boldsymbol{x}}) + \lambda_2 \mathcal{L}_{\text{logMAG}}(\boldsymbol{x}, \hat{\boldsymbol{x}}) + \lambda_3 \mathcal{L}_{\text{linMAG}}(\boldsymbol{x}, \hat{\boldsymbol{x}}),$$

where $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$ denote the target and the estimated audio signal.

λ_1, λ_2 and λ_3 denotes the weight coefficient to balance losses, spectral convergence ($\mathcal{L}_{SC}$), log STFT manginuted loss ($\mathcal{L}_{logMAG}$) and linear STFT magnitude loss ($\mathcal{L}_{linMAG}$), which is defined as follows:

$$\mathcal{L}_{SC}(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \frac{\||\,\text{STFT}(\boldsymbol{x})| - |\,\text{STFT}(\hat{\boldsymbol{x}})|\|_F}{\||\,\text{STFT}(\boldsymbol{x})|\|_F}$$

$$\mathcal{L}_{\text{logMAG}}(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \|\log|\,\text{STFT}(\boldsymbol{x})| - \log|\,\text{STFT}(\hat{\boldsymbol{x}})|\|_1,$$

$$\mathcal{L}_{\text{linMAG}}(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \||\,\text{STFT}(\boldsymbol{x})| - |\,\text{STFT}(\hat{\boldsymbol{x}})|\|_1,$$

Discriminative and Generator Loss

The objective of the adversarial loss is to enhance the visual appeal of the output. It is described as a hinge loss that is applied to the discriminator's logits, and it is averaged over multiple discriminators and time. To be more precise, let k $\in \{0, ..., k\}$ be used to identify individual discriminators, where k = 0 refers to the STFT-based discriminator, and k $\in \{1, ..., K\}$ represents the various resolutions of the waveform-based discriminator (we use K = 3 in this study). Let T_k be the number of logits produced by the k-th discriminator over the time dimension. The goal is for the discriminator to classify original versus decoded audio by minimizing the loss.

$$\mathcal{L}_D = E_x\left[\frac{1}{K}\sum_k \frac{1}{T_k}max(0, 1 - D_{k,t}(x))\right] + E_x\left[\frac{1}{K}\sum_k \frac{1}{T_k}max(0, 1 + D_{k,t}(G(x)))\right]$$

while the adversarial loss for the generator is

$$\mathcal{L}_G^{adv} = E_x\left[\frac{1}{K}\sum_{k,t} \frac{1}{T_k}max(0, 1 - D_{k,t}(G(x)))\right]$$

The feature loss is computed by taking the average absolute difference between the discriminator's internal layer outputs for the generated audio and those for the corresponding target audio.

$$\mathcal{L}_G^{feat} = E_x\left[\frac{1}{KL}\sum_{k,l} \frac{1}{T_{k,l}}\sum_t |D_{k,t}^{(l)}(x) - D_{k,t}^{(l)}(G(x))|\right]$$

where $\mathcal{L}$ is the number of internal layers, $D_{k,l}^{(l)}(l \in \{1, ..., L\})$ is the t-the output of layer l of discriminator k, and $T_{k,l}$ denotes the length of the layer in the time dimension.

VQ commitment loss

We add a commitment loss $\mathcal{L}_w$ between the output of the encoder, and its quantized value, with no gradient being computed for the quantized value. For each residual step $c \in \{0, ..., C\}$ (with C depending on the bandwidth target for the current batch), noting z_c the current residual and $q_c(z_c)$ the nearest entry in the corresponding codebook, we define $\mathcal{L}_w$ as

$$\mathcal{L}_w = \sum_{c=1}^{C} \|z_c - q_c(z_c)\|_2^2$$

Total Generator loss

$$\mathcal{L}_G = \lambda_{adv}\,\mathcal{L}_G^{adv} + \lambda_{feat} \cdot \mathcal{L}_G^{feat} + \lambda_{rec} \cdot \mathcal{L}_G^{rec}$$

4.7 Implementation Details

We trained the model for 3000 epochs, with epoch being 5,000 updates with the Adam optimizer with an initial learning rate of 3e±4 for generator and 1e±4 for the discriminator and default parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$ for optimization. The mini-batch size is set to 10 and the whole system is implemented in PyTorch. We set the $\lambda_1=1$, $\lambda_2=1$ and $\lambda_3=0$ for reconstruction loss and $\lambda_{adv}=1$, $\lambda_{feat}=100$ and $\lambda_{rec}=1$ to scale adversarial, feature and reconstruction losses for generator loss.

5.   Evaluation Setup

5.1   Dataset

We train our model on three types of audio content: clean speech, noisy speech, and music, all at a 16 kHz sampling rate. The audio data from [1] were further processed using ffmpeg to transform the given data into a suitable representation i.e. sampling frequency of 16 kHz with 16 bits per sample. In total 20,000 audio files were used to train the model with 7,500 clean speech, 7,500 noisy speech, and 5,000 music. For validation in total 60 audio files were used with 20 from each clean speech, noisy speech, and music.

5.2   Evaluation Metric

To evaluate our model, we conducted both subjective and objective evaluations of the results. For subjective evaluation the MUSHRA[15] test is used, it involves presenting a listener with several audio stimuli, which can be different versions of the same audio signal, or different audio signals altogether. One of the stimuli is designated as the "anchor", which is a high-quality reference signal that serves as a benchmark for comparison. The listener is asked to rate the quality of each stimulus using a numerical scale, typically ranging from 0 to 100, where 0 represents the worst possible quality and 100 represents the best. Ratings were screened to exclude noisy data. For objective evaluation of the model, SCALE-INVARIANT SIGNAL-TO-NOISE RATIO was used. While SI-SNR is primarily used in speech separation and speech enhancement applications, it can also be used as a metric for evaluating audio codecs.

5.3   Baselines

Opus and EVS are set as baselines for comparison with traditional audio codecs. Opus [3] is a popular speech and audio codec widely used for internet communication applications, such as Zoom, Microsoft Teams, and Google Meet, as well as for streaming on YouTube. It supports a range of signal bandwidths from 4 kHz to 24 kHz and bitrates from 6 kbps to 510 kbps. Another versatile codec used for Voice over LTE (VoLTE) is Enhanced Voice Services (EVS) [2], which operates at multiple signal bandwidths from 4 kHz to 20 kHz and bitrates from 5.9 kbps to 128 kbps. Both codecs serve as baselines for comparison with our model, which is designed to enhance audio quality. For comparison with neural audio codecs, Meta's Encodec [9] and Google's SoundStream [37] are used.

6.   Result and Discussion

The developed audio codec was compared with the baselines in different categories which are explained below:-

Subjective and Objective Evaluation

An experiment was conducted using headphones and 50 English speakers to subjectively evaluate the performance of our codec. The evaluation dataset comprised 20 samples of clean, noisy, reverberant speech, and music. Each sample was rated using a crowd-sourced methodology inspired by MUSHRA. The results of the subjective evaluation are shown in Table 6.5. It can be observed that our codec outperformed traditional audio codec(OPUS) at low bitrates. However, our codec was unable to surpass neural audio codecs like Soundstream (by Google) and Encodec (by Meta). For comparison with the Baseline, our model with a high MUSHRA score was selected and the performance of our model with different additional components can be seen in Table 6.1. In order to further improve the quality of the decompressed audio, a Discriminator was used to remove the artifacts from the reconstructed audio signal as shown in Table 6.2. Table 6.3 shows that our model's SNR score is far less than others which makes the MUSHRA the main evaluation metric for the analysis of our model. The poor SI-SNR score is mainly because it is primarily used in speech separation and speech enhancement applications.

Openness

Also, there are no fully open-source neural audio codecs available on the internet. Table 6.4 shows the open-source state of different neural audio codecs on the internet.

Table 6.1: MUSHRA(↑) scores of our models with different additional components. Performance of our model with different additional components at different bitrates along with the model complexity. Here rrdb represents the Residual-in-Residual-Dense Block from [23] which replaces the normal residual block of the model but increases complexity and jitter is the time jitter regularization for the robustness of quantized space.

Model	3 kbps	6 kbps	12 kbps	24 kbps	Params
Ours	15	32	35	38	6.27M
Ours + lstm	20	40	45	50	14.67M
Ours + lstm + dropout(0.2)	**25**	40	**52**	**55**	14.67M
Ours + lstm + dropout(0.5)	15	16	15	17	14.67M
Ours + lstm + dropout(0.2) + rrdb	22	**44**	50	50	30.5M
Ours + lstm + dropout(0.2) + jitter	10	12	11	10	14.67M

Table 6.2: MUSHRA(↑) scores our model with and without discriminator for 6 kbps.

Model	Speech	Music
without disc	29	32
with disc	40	47

Table 6.3: Comparision of SI-SNR(↑) scores.

Model	Speech
OPUS	2.45
EVS	1.89
ENCODEC	6.67
SoundStream	6.5
Ours	1.2

Table 6.4: Openness of different models compared to ours.

Codec	Code	Model	Paper	Reusability
SoundStream	✗	✗	✓	✗
ENCODEC	✗	✓	✓	✗
Ours	✓	✓	✓	✓

Table 6.5: MUSHRA(↑) scores comparison of our's best model from Table 6.1 with different state-of-the-art traditional and neural audio codec.

| Model	Bandwidth	Speech	Music
Reference	-	$95.5_{\pm1.6}$	$93.2_{\pm2.5}$
OPUS	6.0 kbps	$30.1_{\pm2.8}$	$20.6_{\pm5.8}$
OPUS	12.0 kbps	$76.5_{\pm2.3}$	$77.8_{\pm3.2}$
EVS	9.6 kbps	$84.4_{\pm2.5}$	$89.9_{\pm2.3}$
Lyra-v2	3.0 kbps	$53.1_{\pm1.9}$	$69.3_{\pm3.3}$
Lyra-v2	6.0 kbps	$66.2_{\pm2.9}$	$75.7_{\pm2.6}$
ENCODEC	1.5 kbps	$49.2_{\pm2.4}$	$68.2_{\pm2.2}$
ENCODEC	3.0 kbps	$67.0_{\pm1.5}$	$89.6_{\pm3.1}$
ENCODEC	6.0 kbps	$83.1_{\pm2.7}$	$92.9_{\pm3.1}$
ENCODEC	12.0 kbps	$90.6_{\pm2.6}$	$91.8_{\pm2.5}$
SoundStream	3.0 kbps	$67.0_{\pm1.5}$	-
SoundStream	6.0 kbps	$80.0_{\pm1.5}$	-
SoundStream	12.0 kbps	$82.0_{\pm1.5}$	-
Ours	3.0 kbps	25.0	30.0
Ours	6.0 kbps	40.0	47.0
Ours	12.0 kbps	50.0	52.0
Ours	24.0 kbps	50.0	55.0
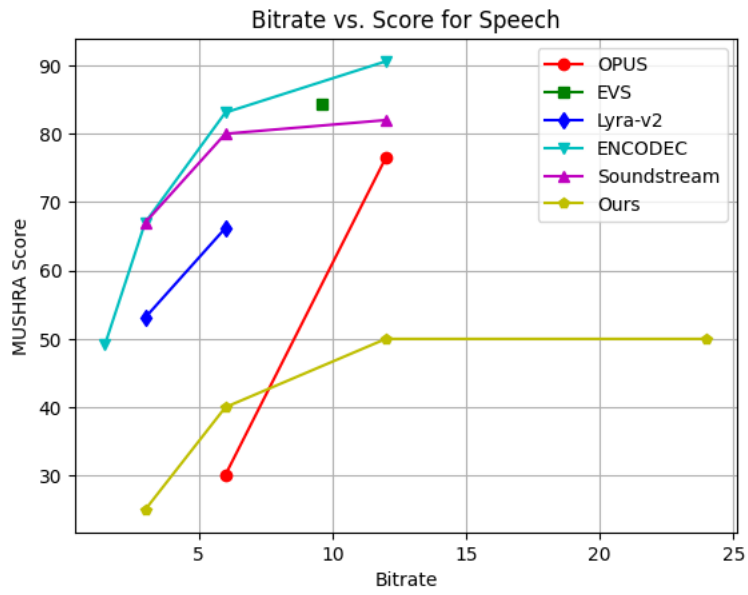
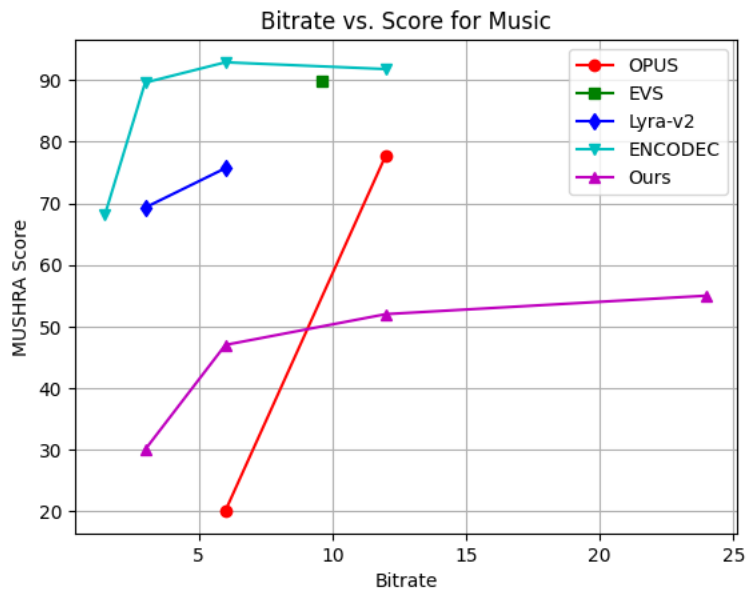Figure 6.1: Ours vs state-of-art codecs for Speech

Figure 6.2: Ours vs state-of-art codecs for Music

20

Latency and Real Time Factor

Latency refers to the delay between the time an audio signal enters the codec and the time it is processed and output. In other words, it is the time it takes for the codec to encode or decode the audio signal. We profiled our model on a single thread of an intel i7-10870H CPU and Nvidia RTX 3060 GPU at 6 kbps while other models' latency and RTF values are taken from Table 5 of [9] where these models are evaluated on a single thread of a MacBook Pro 2019 CPU. The calculation of latency involves the time taken by ffmpeg to transform audio to 16-bit 16000 hz audio plus the model's encoding and decoding speed. The ffmpeg nearly takes about 6 ms to convert the audio which increases the latency of the codec. The real-time factor is here defined as the ratio between the duration of the audio and the processing time so that it is greater than one when the method is faster than real-time. The calculation of latency and real-time factor is shown in Table 6.6. In this analysis, our codec out-perform the other audio codec mainly due to its smaller model size.

Table 6.6: Latency(↓) and RTF(↑) of our model and ENCODEC, Lyra v2, SoundStream, OPUS. All models are evaluated at 6 kbps.

| | | Real Time Factor | |
Model	Latency	*Enc.*	*Dec.*
Lyra v2 (32 kHz)	-	27.4	67.2
ENCODEC 24 kHz	13 ms	9.8	10.4
ENCODEC 48 kHz	1 s	6.8	5.1
SoundStream 24 kHz	13 ms	-	-
OPUS	26.5 ms	-	-
Ours 16 kHz (cpu)	12 ms	55.56	111.11
Ours 16 kHz (gpu)	**11 ms**	**83.33**	**125**

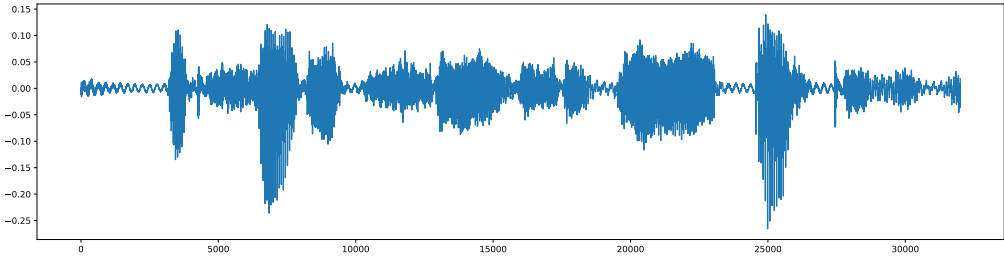**Original and Reconstructed audio waveform**

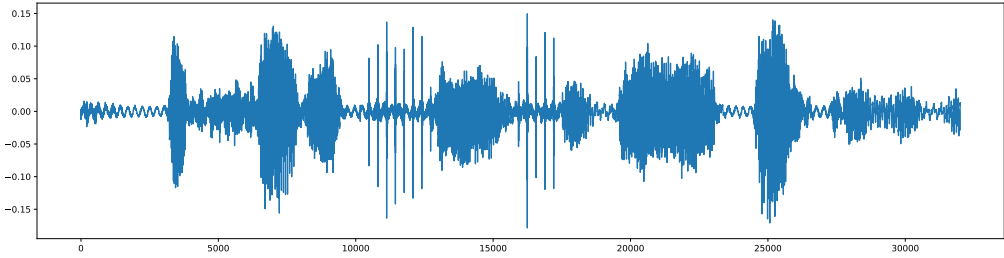Figure 6.3: Original Waveform.

Figure 6.4: Reconstructed Waveform.

7.   Conclusion

In conclusion, this project highlights the potential of deep learning-based audio codecs in enhancing compression efficiency and improving audio quality. The subjective evaluation conducted in this study shows that our Neural Audio Codec performs comparably to traditional audio codecs, Opus and EVS, at low bitrate despite being a relatively new approach to audio compression. By releasing open-source code and pre-trained models, this study provides a valuable resource for further research and improvement. Moreover, with the endless possibility of further advancements in deep learning-based audio synthesis and representation learning, there is a significant potential for improving audio codecs' efficiency and quality. The future of audio compression seems promising, and the development of deep learning-based models can play a pivotal role in it.

References

[1] Muscan dataset. `https://github.com/microsoft/multimodal-speech-corpora`, 2020.

[2] EVS Codec. `https://www.etsi.org/technologies/evs-codec`, 2022. [Online].

[3] Opus interactive audio codec. `https://opus-codec.org/`, 2022. [Online].

[4] Arindam Biswas and Dai Jia. Audio codec enhancement with generative adversarial networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 356–360, 2020.

[5] Karlheinz Brandenburg. MPEG audio coding: Basics and state of the art. *Journal of the Audio Engineering Society*, 47(10):789–814, October 1999.

[6] Jan Chorowski, Ron J Weiss, Samy Bengio, and Aäron van den Oord. Unsupervised speech representation learning using wavenet autoencoders. *arXiv preprint arXiv:1901.08810*, 2019.

[7] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[8] Chris Donahue, Julian McAuley, and Miller Puckette. Adversarial audio synthesis. *arXiv preprint arXiv:1911.12616*, 2019.

[9] Alexandre Défossez, Jade Copet, Gabriel Synnaeve, and Yossi Adi. High fidelity neural audio compression. *arXiv preprint arXiv:2210.13438*, 2022.

[10] Yifei Fang, Rui Hu, Lei Xie, and Chin-Hui Lee. Vq-vae based multi-band spectrogram modeling for voice conversion. In *Proceedings of Interspeech*, 2020.

[11] Cristina Garbacea, Aäron van den Oord, Yazhe Li, Felicia S C Lim, Alejandro Luebs, Oriol Vinyals, and Thomas C Walters. Low bit-rate speech coding with vq-vae and a wavenet decoder. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 735–739, 2019.

[12] Cătălin Gârbacea, Shiyu Liao, Alexei Baevski, and Geoffrey Zweig. A vector quantized variational autoencoder approach to generative speech modeling. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6455–6459, 2019.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pages 630–645, 2016.

[14] Fraunhofer IIS. Mp3: A digital audio coding format for consumer audio storage and transfer. *Journal of the Audio Engineering Society*, 45(9):789–798, Sep. 1997.

[15] ITU-R. Method for the subjective assessment of intermediate quality level of coding systems. Technical report, International Telecommunications Union, 2001.

[16] Vineeth Jayashankar, Xiaohan Shen, and Shrikanth Narayanan. On the role of quantization in autoencoder-based speech compression. *arXiv preprint arXiv:2202.00908*, 2022.

[17] Xiaoyu Jiang, Shenghui Peng, and Yujia Yan. A new design for a low-bitrate speech codec based on vqvae. *arXiv preprint*, 2022.

[18] N. Kalchbrenner, E. Elsen, K. Simonyan, S. Noury, N. Casagrande, E. Lockhart, F. Stimberg, A. van den Oord, S. Dieleman, and K. Kavukcuoglu. Efficient neural audio synthesis. *arXiv preprint arXiv:1802.08435*, 2018.

[19] Sharath Kankanahalli. End-to-end optimized speech coding with deep neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2521–2525, 2018.

[20] W Bastiaan Kleijn and Jorma Laaksonen. Straight-to-audio deep residual network for efficient spectral envelope estimation from bandlimited speech signals. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5304–5308, 2018.

[21] W Bastiaan Kleijn and Jorma Laaksonen. Deep neural network-based spectral envelope compression for efficient wideband telephony. *IEEE Transactions on Audio, Speech, and Language Processing*, 29:396–407, 2021.

[22] WB Kleijn, FS Lim, A Luebs, J Skoglund, F Stimberg, Q Wang, and TC Walters. Wavenet based low rate speech coding. In *IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 676–680, 2018.

[23] Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. *arXiv preprint arXiv:2010.05646*, 2020.

[24] Kundan Kumar, Kumar, Rithesh Kumar, Thibault De Boissiere, Lucas Gestin, Wei Zhen Teoh, Jose Sotelo, Alexandre de Brébisson, Yoshua Bengio, and Aaron C Courville. Melgan: Generative adversarial networks for conditional waveform synthesis. *arXiv preprint arXiv:1910.06711*, 2019.

[25] Jaehoon Lim, Junho Sung, and Changho Kim. High-quality and efficient generative models for audio synthesis using self-attention and dilated convolutions. *arXiv preprint arXiv:2006.04922*, 2020.

[26] Shigeru Morishima, Hiroshi Harashima, and Yoshinori Katayama. Speech coding based on a multi-layer neural network. In *IEEE International Conference on Communications, Including Supercomm Technical Sessions*, pages 429–433, 1990.

[27] Jan Skoglund and Jean-Marc Valin. Waveform coding using deep generative models. In *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 245–249. IEEE, 2019.

[28] Thomas Sporer, Karlheinz Brandenburg, Bernhard Grill, Marina Bosi, and Gerhard Stoll. The aac audio coding standard: History and technical overview. *Journal of the Audio Engineering Society*, 52(6):618–631, June 2004.

[29] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[30] Marco Tagliasacchi, Yunpeng Li, Karolis Misiunas, and Daniel Roblek. Seanet: A multimodal speech enhancement network. In *Interspeech*, pages 1126–1130, 2020.

[31] Jean-Marc Valin and Jan Skoglund. Neural speech compression for low-bitrate communications. *arXiv preprint arXiv:1911.10436*, 2019.

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

[33] Aäron van den Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George van den Driessche, Eddie Lockhart, Luis Cobo, Florian

Stimberg, Nicola Casagrande, Dominik Grewe, Sander Noury, Sander Dieleman, Eric Elsen, Nal Kalchbrenner, Heiga Zen, Alex Graves, Helen King, Tom Walters, Dmitry Belov, and Demis Hassabis. Parallel wavenet: Fast high-fidelity speech synthesis. In *Proceedings of the 35th International Conference on Machine Learning*, pages 3918–3926, 2018.

[34] Aäron Van Den Oord, Oriol Vinyals, and koray kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, volume 30, pages 6306–6315, 2017.

[35] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8798–8807, 2018.

[36] Ryuichi Yamamoto, Eunwoo Song, and Jaehyeon M Kim. Probability density distillation with generative adversarial networks for high-quality parallel waveform generation. *arXiv preprint arXiv:1904.04472*, 2019.

[37] Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. Soundstream: An end-to-end neural audio codec. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2021.

[38] Meng Zhang and Jia Chen. A survey on deep learning for audio signal processing. *IEEE Transactions on Neural Networks and Learning Systems*, 31(9):3369–3413, 2020.

[39] Kai Zhen, Jongmo Sung, Mi Suk Lee, Seungkwon Beack, and Minje Kim. Cascaded cross-module residual learning towards lightweight end-to-end speech coding. In *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, pages 1128–1132, Graz, Austria, September 2019.

8. Appendices

Appendix A: Vector Quantization Compression Mechanism

- The encoder produces one embedding for e.g. 0.1s of audio

- A vector of length 3 encode in float16 use 3 * 16= 48 bits
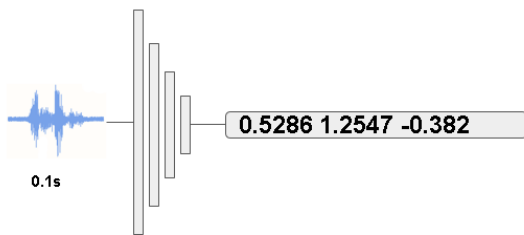
- Per second: 48 * 10 = 480 bits

Figure 8.1: Step I

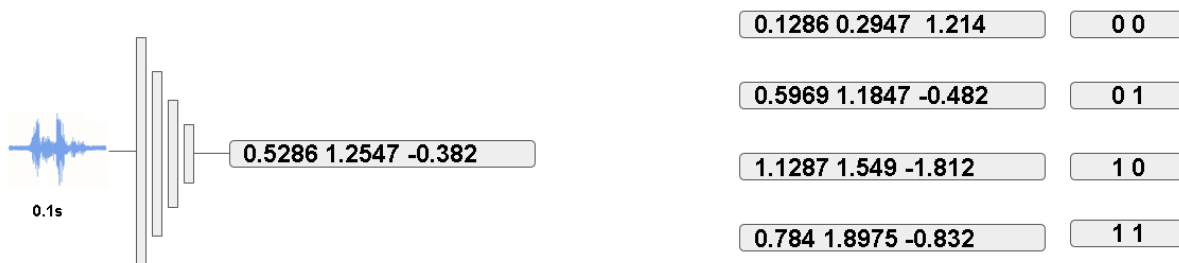- A vector quantizer learns a fixed "dictionary" of template vectors (called code-words)

Figure 8.2: Step II

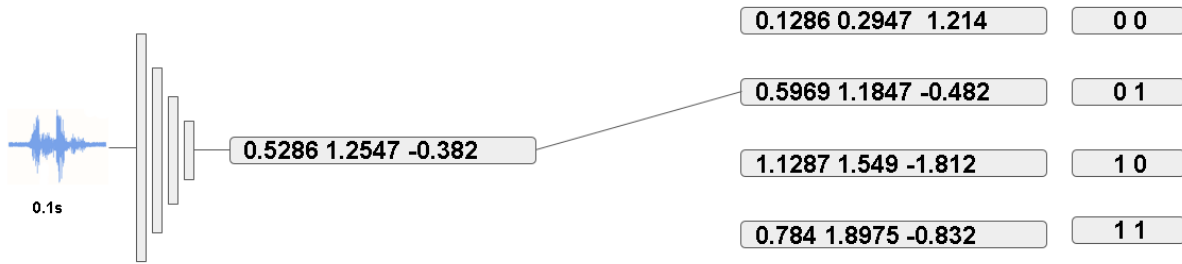- We replace the embedding by its closest neighbor in the dictionary

Figure 8.3: Step III

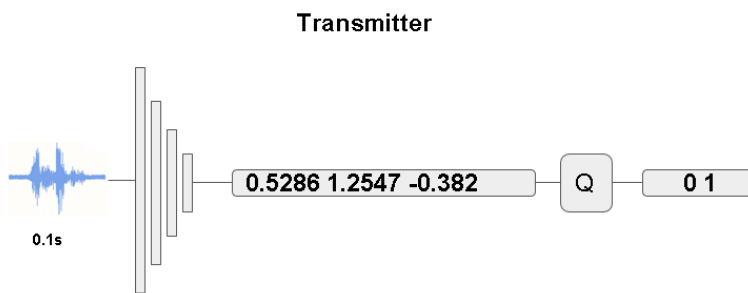- Now we only need to transfer 2 bits (the index) to the receiver

Figure 8.4: Step IV

- The receiver also has the codebook and uses it to retrieve the vector corresponding to the index
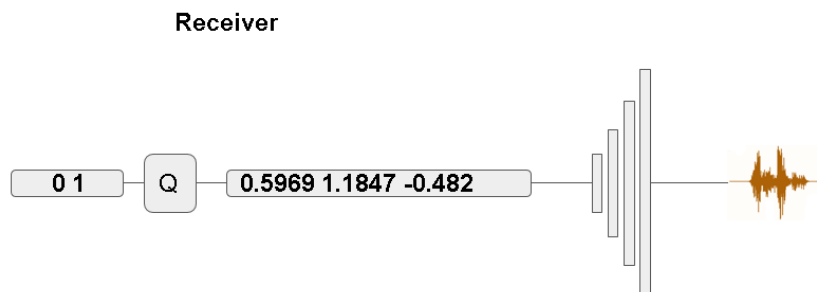

Figure 8.5: Step V

Appendix B: Custom File Format (nac)

- Byte order : network big endian

- Header format (9 Bytes)

    - 3 bytes: magic string

    - 1 byte : version number

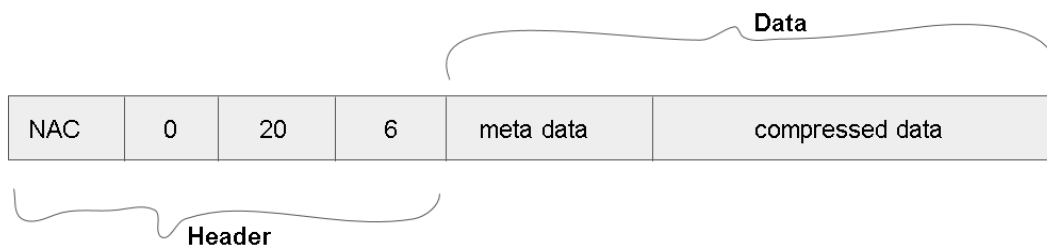   - 4 byte: metadata length

    - 1 byte: bit rate

| NAC | 0 | 20 | 6 | meta data | compressed data |

Figure 8.6: custom file format (.nac)