A Comparative Study of Audio Compression Algorithms and Their Effects on Automated Speech Recognition Accuracy

Ian Low

Master of Science
School of Informatics
University of Edinburgh
2023

Abstract

This study investigates the specific mechanisms of lossy audio codec compression techniques and their impact on speech intelligibility, assessed using Automated Speech Recognition (ASR) platforms and measured via Word Error Rates (WER). We present a comprehensive review of the development of audio codecs, detailing their compression methodologies and anticipated performance efficiencies. Experiments were conducted using ASR systems to empirically measure the codecs' performances. Our findings reveal a clear relationship between the intricacy of encoding methods and speech intelligibility, notably under low bit rate scenarios. In testing conditions involving background noise, speech intelligibility was surprisingly improved through the use of codecs, indicating that these codecs possess noise suppression capabilities. By leveraging ASR systems for speech quality assessment, we offer a reliable and consistent evaluation method, establishing ASR platforms as novel evaluation tools for upcoming research in compression and speech processing.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Ian Low*)

Acknowledgements

I would like to extend my heartfelt gratitude to Dr. John Longley for his invaluable guidance and unwavering patience throughout this project. Our weekly meetings and his consistent encouragement have been instrumental in shaping and refining this research.

Table of Contents

Chapter 1

Introduction

In today's digital age, many of our spoken words and conversations are captured and stored in digital formats like podcasts, audio recordings, and voice messages. While this digitisation has made information more accessible and manageable, it has also created challenges in storing and transmitting large volumes of speech data. Compression, which reduces data size while trying to retain its quality, has become essential in addressing these challenges. However, there are trade-offs. While lossy compression techniques can achieve higher compression ratios, they might also compromise the intelligibility of the speech due to information loss.

A significant amount of research has explored how compression affects our ability to understand speech in terms of evaluating the intelligibility. Yet, there is a notable gap when it comes to understanding how Automated Speech Recognition (ASR) systems fare under similar evaluations. ASR systems, now a regular part of our lives, might handle compressed speech differently than humans. This project aims to understand how compression techniques impact the accuracy of ASR systems in transcribing speech.

Two key developments not only enable the foundation of this study but also underscore its relevance. First, ASR systems are not only widely used, but their services have also become extensively available, paving the way for our experiments. Second, due to more recent technological advancements, today's ASR systems produce reliable results that allow for meaningful and valid assessments. Their refined capability to transcribe compressed audio might even exceed human abilities in specific instances. The findings from this research could highlight potential challenges with certain codecs and provide direction for the development of codecs that are ideally suited for ASR applications.

This report is structured as follows. We will first review the concepts and principles of audio data compression in Chapter 2. The chapter includes a survey of several com-

pression techniques, as well as a basic tutorial of a critical speech model used in many of the techniques. In Chapter 3, we review our experimental design, considerations, scope and set up. The results of our experiments are then shown in Chapter 4. Important trends and topics that emerged from our results are discussed and further analysed in Chapter 5, before we end the report with our concluding remarks in Chapter 6.

Chapter 2

Background

This chapter aims to illuminate some of the underlying principles of audio data compression. We surveyed algorithms and techniques that are commonly used in the industry, and assembled a cohesive account of how these techniques build upon each other to achieve higher degrees of sophistication and efficiency. We regard this synthesis of concepts from multiple sources and perspectives as a significant contribution to the project. This narrative not only presents the landscape of digital audio encoding but also provides clarity and insights into the evolution of techniques and innovations that have shaped this field.

In the realm of digital audio, compression is performed by algorithms known as '*codecs*', an abbreviation for 'coder-decoder'. These are systems designed not only to perform compression during the encoding stage, but also to decode and reconstruct the data for playback at the receiving end. Many of these techniques described here are used in the codecs that will be visited in our experiments.

The primary objective of compression, applicable to both speech and general audio, is to decrease *redundancy* in the signal. This reduces the bandwidth of the audio stream while retaining acceptable or minimal loss to its quality, as perceived by listeners. Audio signals can exhibit various forms of redundancy. For instance, *temporal redundancy* refers to a signal's tendency to have adjacent samples that are either similar to each other, or are repeated. On the other hand, there is *spectral redundancy* at frequencies that exceed the typical human hearing threshold of 20kHz. By understanding how listening works, it is possible to design techniques that preserve the most critical characteristics of a signal, discard unusable information, and handle repetition in ways that are more data efficient.

Broadly, speech compression techniques fall into two categories—waveform-based

and parametric-based compression. The former is typically concerned with encoding the individual samples themselves, whereas the latter requires some signal analyses to extract deeper representations of the same signal. Modern codecs will usually employ a mixture of several techniques spanning both classes.

We will also examine the role of psychoacoustics in audio data compression, which is the science of how humans perceive sound, and how this understanding contributes to more effective compression strategies.

2.1 Waveform Compression

In this section, we will start by understanding Linear Pulse-code Modulation, a standard 'non-compressive' technique for representing a digital audio signal, before examining ways of improving its efficiency.

2.1.1 Linear Pulse-code Modulation (Linear PCM)

When an analog signal is converted into a digital form, the amplitude of the signal is sampled at uniform intervals determined by the sampling frequency. Where the analog signal's amplitude spanned a continuous range, it now also has to be quantised into discrete levels so that it can be digitally represented.

Linear PCM represents the most straight-forward way of doing this quantisation - the amplitudes are rounded to discrete steps determined by the bit depth. These steps are evenly spaced apart, hence the *linear* term. *Quantisation noise* is inevitably introduced from the rounding off of the original signal to the nearest step. A higher bit depth results in more quantisation steps, lower quantisation noise, and better audio quality. For example, the Compact Disc Digital Audio (CDDA), which is the standard format for audio CDs, uses 16-bit Linear PCM for encoding audio data.

Because Linear PCM mainly concerns itself with representing the waveform as accurately as possible at the specified bit depth and sampling rate, and does not involve any means to compress or otherwise improve on its efficiency, it can generally be viewed as being an uncompressed format. Linear PCM is also not considered a compression technique, but rather a technique for audio recording and representation, forming the basis upon which subsequent compression techniques can be applied.

2.1.2 Non-Linear Pulse-code Modulation

Human hearing is more sensitive to quieter sounds than to louder sounds. We are thus able to better distinguish fluctuations at the quieter end, than we can for pertubations at the louder end. This is similar to a logarithmic response, where sensitivity declines non-linearly with increasing amplitudes.

Non-linear PCM codecs like the G.711 A-law and G.711 μ-law [21], both standardised by the International Telecommunication Union (ITU), take advantage of this. By applying a logarithmic scale to the quantisation levels, these codecs provide a greater resolution for quieter sounds, which aligns with our sensitivity to these sounds. Conversely, a lower resolution is used for louder sounds, as our hearing is not as sensitive at these levels anyway.

This strategy reduces quantisation noise in quieter, more sensitive parts of the audio signal, while any increased noise in the louder sections is less perceivable. Overall, this achieves a better audio experience to the listener and presents an improvement in terms of bandwidth efficiency. The improvement in efficiency enables strategic reductions in bandwidth by reducing the encoding bit depth without significantly compromising the audio experience.

2.1.3 Differential Pulse-code Modulation (DPCM)

Although the amplitudes of audio signals can span the entire range within the limits, successive samples of the signal are more likely to be similar rather than swinging from one limit to its polar opposite. Differential PCM exploits this property by encoding the differences in amplitude instead of the absolute values for each sample. This allows DPCM to sequentially map the most likely amplitudes with a lower bit depth. In this way, DPCM can be more efficient in its encoding when applied to audio signals.

Adaptive Differential Pulse Code Modulation (ADPCM) takes DPCM a step further by varying the size of the quantisation step differences. The step sizes are dynamically adjusted depending on the expected size of the fluctuations, operating on the expectation that the size of the current fluctuation will be similar to that of recent fluctuations. While this requires some additional processing to analyse past samples and make predictions, ADPCM codecs, such as G.726 [22], standardised by the ITU, can reduce overall quantisation noise with the same bit depth, making it a more effective form of compression compared to DPCM.

2.2 Parametric Compression

PCM and its variants such as DPCM and ADPCM are techniques used in waveform representation and compression. They work by sampling the waveforms at regular intervals and quantising them, allowing a reproduction of the original waveform that is as accurate as the bit depth and sampling rate allow. In the following sections, we will shift our focus to a different paradigm - *parametric compression*. Instead of encoding raw waveform data at every interval, parametric compression captures characteristics of the signal such as pitch, gain, and spectral information, using certain models of sound production. This model-based approach allows parametric compression to achieve higher compression ratios than waveform-based methods, with minimal losses to perceived audio quality.

The predominant model adopted in many compression techniques is the source-filter model. We begin this section by first understanding the relationship between the time and frequency domains, which is foundational to the source-filter model.

2.2.1 Time and Frequency Domains

The waveforms that we have discussed earlier are the signals' representation in the time domain. A waveform is a functions of time, showing how the signal's amplitude changes as time progresses. Conversely, the frequency domain reveals how the signal's energy is distributed across different frequencies. This distribution is often referred to as the *spectrum*, and it represents the signal as if it were a combination of sine waves of various frequencies, each with its own amplitude. By understanding the spectrum, we can identify the constituent parts of a complex signal, aiding us in tasks like recognising harmonics, fundamental frequencies, or sources of noise.

The connection between the time and frequency domains is made through the Fourier transform and its inverse [16]. The Fourier transform, given by the equation $F(f) = \int_{-\infty}^{\infty} f(t) \cdot e^{-j2\pi ft}\, dt$, breaks down a complex waveform into the constituent sinusoids, moving from the time domain to the frequency domain. The inverse transform does the opposite, recombining those sinusoids into the original waveform.

Consider the relationship between pure sinusoidal tones and their representations in the frequency domain shown in Figure 2.1. In the frequency domain, pure tones manifest as single spikes, each positioned at their respective frequencies and with heights corresponding to their amplitudes. The sum of individual waveforms results in a more intricate waveform in the time domain. However, analyzing this composite signal

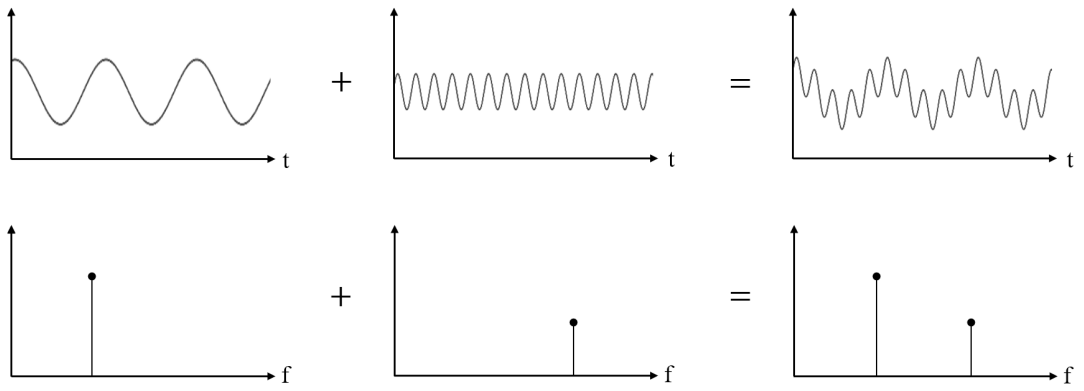

Figure 2.1: Waveforms (top) and their representations in the frequency domain (bottom). In this example, a pure tone of lower frequency and higher amplitude (left) is added to another pure tone of higher frequency and lower amplitude (centre) to form a more complex waveform. Equivalently, complex waveforms such as the one on the right can be decomposed into its constituent sinusoids via the Fourier transform.

in the frequency domain via the Fourier transform reveals its individual constituent tones, given by distinct spikes, which is simpler and more intuitive to interpret.

Converting a time-based signal to the frequency domain typically involves selecting a window (or frame) of a certain length, setting the time boundaries of the transformation. Because the Fourier transform integrates the signal over this time period, the signal is assumed to be stationary within the window. For speech signals, it is common to use window lengths of between 20ms to 30ms, which correspond to the quasi-stationary nature of speech.

This windowing approach ensures that we are analyzing a finite segment of the signal, providing a snapshot of its frequency content during that interval. Using shorter windows allows for better frequency resolution, enabling more detailed insights into the signal's components. Conversely, longer windows offer a broader view of the signal but would reduce the frequency resolution, making it more difficult to discern specific components.

Understanding the interplay between the time and frequency domains is essential in signal processing. The time domain provides a view of 'when' things are happening in a signal, giving a complete picture at every instant. Conversely, the frequency domain tells you 'what' frequencies are present by dissecting the signal into its basic frequency components. With an understanding of how signals can be represented in both time and frequency domains, we will now build on these principles and examine the source-filter model, which plays a critical role in speech compression techniques.

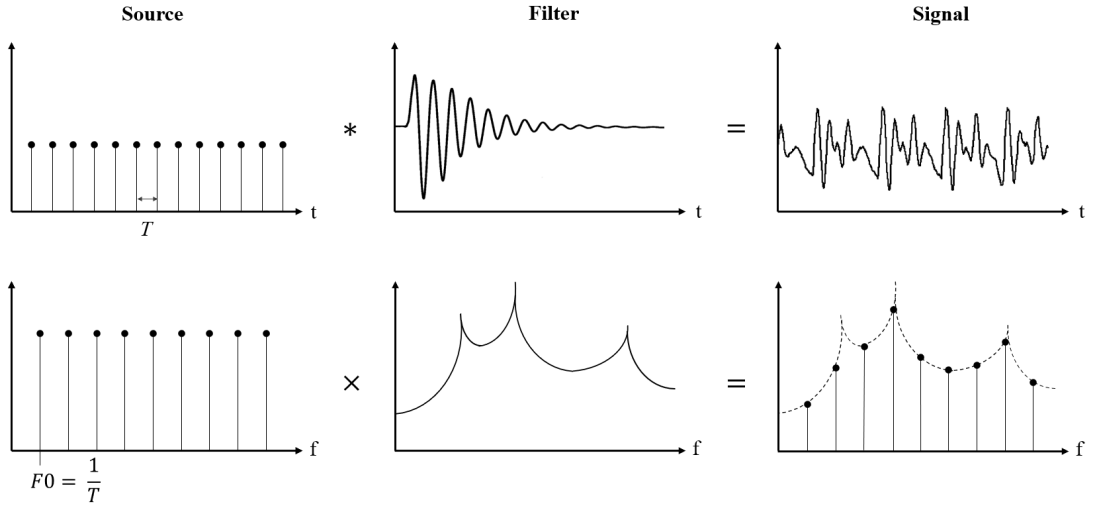

Figure 2.2: Interaction of source and filter functions in the time domain (top) and frequency domain (bottom). The interaction is characterised by the convolution operation in the time domain, and multiplication in the frequency domain, which results in the output spectrum (bottom right) for a frame of speech.

2.2.2 Source-filter Model

Speech possesses unique characteristics that make it distinct from other types of sounds. It is a complex process involving the production of sounds by the *vocal folds*, which then propagate along the *vocal tract*, where it is shaped by the curvature and length of the tract as well as the articulators, such as the tongue, palate, teeth, and lips. The source-filter model approximates this understanding by describing speech production as involving two main processes: the generation of a *source* signal, and the modification of that signal through a *filter* [28].

Figure 2.2 shows a simplified version of the two processes at work. The source signal, which mimics the sound produced by the vocal folds, can be approximated as a uniform pulse train with fixed period T. If we were to do the Fourier transform, the source signal can be equivalently represented as a combination of sinusoids with frequencies that are multiples of the *fundamental frequency* $F0$, or $\frac{1}{T}$. These evenly spaced frequency components (not to be confused with a pulse train) are the *harmonics* of the fundamental frequency.

The source signal is then modified by the filter function, which emulates the resonant structures of the vocal tract. In the time domain, this interaction is characterised by the so-called *convolution* operation. When performed in the frequency domain however, the relationship is frequency-wise multiplicative and easier to compute.

This multiplicative relationship between the two functions allows us to consider speech frames as a product of their source and filter components. Many of the subsequent techniques described aim to separate aspects of the source and filter functions and parameterise them as part of their compression encoding.

2.2.3 Linear Predictive Coding and Excitation Signal

In Linear Predictive Coding (LPC), the vocal tract's filter function is approximated as an infinite impulse response (IIR) filter. An IIR filter can be understood through its difference equation representing the relationship between the input and the output of the filter.

The difference equation of an IIR filter is given by

$$y[n] = b_0 \cdot x[n] + b_1 \cdot x[n-1] + \ldots + b_M \cdot x[n-M] - a_1 \cdot y[n-1] - \ldots - a_N \cdot y[n-N]$$

where $y[n]$ is the output of the filter at time n, $x[n]$ is the input to the filter at time n, M and N are the number of past input and output samples, respectively, used for the filter's history, and a_i and b_i are some coefficients used to weigh the outputs and inputs. The equation shows that the output depends not only on present and past inputs, but also the past outputs, which are controlled by the a_i coefficients. This effectively sets up a feedback mechanism across outputs, whereby the current output is recursively linked to all past outputs. It is through this relationship that the filter gets its 'infinite' nature.

There are two reasons for this approximation. Firstly, the vocal tract resembles a resonant tube with resonant frequencies (or formants) indicated by the peaks of the filter function. Similarly, an IIR filter is also capable of possessing several resonant frequencies, also known as the filter's poles [29]. Secondly, the output of an IIR filter is dependent not just on the current input sample, but also on the previous outputs. This property is also observed in speech signals—within a short time frame, speech samples are highly correlated.

During LPC analysis, the current sample is assumed to be a linear combination of a fixed number of past samples. The coefficients of this combination, often referred to as LPC coefficients, are computed in such a way that minimises the prediction error [18]. These coefficients, akin to the a_i coefficients of the idealised IIR filter, also capture the essential information about the vocal tract's filter function, including the resonant frequencies or formants, which are crucial features of speech. Through the assumptions made above, LPC is thus able to parametrically encode the spectral envelope of speech signals, providing a compact yet informative representation.

With the filter component of the source-filter model taken care of, we can now focus on the source component to complete the signal reconstruction. Passing the original signal through the inverse of the newly obtained filter function will yield a residual signal. This residual, often referred to as the *excitation signal* is the driving signal behind the speech production and is analogous to the aforementioned source.

The simplest way of representing the excitation signal is to assume it to be a uniform pulse train with fixed amplitude and frequency. The signal would thus exhibit perfect harmonics, not unlike those shown in Figure 2.2. To parameterise this, the residual signal will be analysed to determine the fundamental frequency and its appropriate gain. However, not all speech can be modelled in this way. Unvoiced segments of speech, such as 's' or 'f' sounds, are more akin to random noise than a harmonious pulse train. In these cases, the excitation signal is modelled as white noise, which lacks any clear periodic structure. Therefore, depending on whether voicing is detected, the excitation signal is either a pulse train of a specified frequency, or white noise.

Despite its simplicity, this combination of the LPC filter and a basic excitation signal has proven to be effective. It has been used in the LPC-10 codec, a standard for low bit rate voice communication [30].

2.2.4 Improving the Post-LPC Excitation Signal

LPC is a useful technique to represent spectral information like resonant structures, but itself is unable to determine the excitation signal. Post-LPC, a pulse train is the simplest implementation of an excitation signal, but nevertheless results in unnatural sounding voices with a 'mechanical buzz' when the speech is resynthesised at the decoding end. This inadvertently affects the intelligibility and perceived quality of the output.

The following subsections explore how, after LPC is performed, the residual signal can be better approximated other than by using a uniform pulse train, providing it with more complexity and nuance.

2.2.4.1 Regular Pulse Excitation-Long Term Prediction (RPE-LTP)

The post-LPC residual signal in RPE-LTP is analysed in two stages to obtain more parameters that can encode a richer excitation signal.

First, a long term prediction (LTP) analysis is performed to calculate the overall periodicity and gain of the signal. This is similar to determining the pitch and gain of a uniform pulse train as if it were the excitation signal. However, the driving signal is not

assumed to be a uniform pulse train, and further analysis is done in a second stage to determine more nuances.

In the RPE stage, an inverse LTP filter is constructed using the results from the LTP analysis. Applying the filter on the signal removes the longer-term periodicity and gain that LTP had predicted, yielding another residual *innovation signal* that exhibits more short-term unpredictability and variability. RPE then encodes this residual by segmenting it into subframes of equal length, and assigning excitation pulses that best represent each subframe. The result is a series of pulses that have non-uniform positions, amplitudes and polarities. Although the pulses are non-uniform, they each represent a *regularised* subframe, giving RPE its name. While RPE is limited in expressing fuller complexities of the excitation signal, it nevertheless captures some of its dynamic variations that a uniform pulse train wholly fails at.

The GSM 06.10 codec (commonly known as GSM Fullrate) uses RPE-LTP in conjunction with LPC to encode speech signals [11]. It was developed by the European Telecommunications Standards Institute (ETSI) for use in the Global System for Mobile Communications (GSM).

2.2.4.2 Code Excited Linear Prediction (CELP)

Another way of handling the excitation signal can be seen in CELP, which also employs a two-stage approach. The post-LPC residual is first analysed against an *adaptive codebook*, which stores a buffer of the previous samples. The analysis applies different delays on the buffer until a shifted block of past samples best matches the current analysis block. The delay is recorded as the pitch lag, and the signal gain determined [18]. This is similar to the LTP stage in obtaining the overall pitch and gain of the analysis frame. After subtracting the best matching delayed block from the input, we are left with yet another residual signal. This final signal is matched against a *fixed codebook* of predefined innovation signals with pulses in varied positions, amplitudes, and polarities. This step captures the finer detail in the signal that could not be predicted in the adaptive stage, and is analogous to the RPE stage or RPE-LTP.

Using a system of codebooks, CELP relies more on encoding codebook indexes and less on speech parameters. Indexes can be thought of as generally requiring less data to store and transmit. Given a wide repertoire of coded candidate signals, CELP is able to achieve better efficiency in its compression. Codecs such as Speex, developed by the Xiph.Org Foundation, utilise CELP in speech encoding [37].

2.2.4.3 Algebraic Code Excited Linear Prediction (ACELP)

The utilisation of codebooks can yet be further enhanced. In contrast to CELP, which uses fixed codebooks, codecs like the Adaptive Multi-Rate (AMR) implement algebraic codebooks [12]. In this scenario, excitation signals are calculated on-the-fly using predefined algebraic rules and functions. These rules have been designed to capture the expected characteristics, patterns, and variations in speech signals. ACELP thus encodes algebraic parameters for the excitation signal, whereas CELP encodes signal indexes.

ACELP's algebraic rules are tailored to speech signals and thus align closely with the nuances of speech. It is thus more effective than CELP's generalist codebook at handling speech applications, given the same bit depth or bit rate.

Importantly, ACELP *calculates* the best-fitting algebraic parameters during analysis, whereas CELP is limited by a *search* for the best-matching signal. This difference also allows ACELP to express more dynamism in the excitation signal than CELP typically would allow with its finite codebook.

Overall, this results in ACELP encoding a closer representation of the original signal, lower quantisation noise and distortion, and better speech quality than CELP.

2.2.4.4 Conjugate-Structure ACELP (CS-ACELP)

Algebraic codebooks can also be adapted for computational efficiency. In CS-ACELP, two sub-codebooks are utilised instead of the one in ACELP. The post-LTP excitation signal is divided into two tracks, which can be viewed as complementary sub-samplings of the original residual signal. Each track's signal is then analysed against its corresponding sub-codebook to find the best match. All told, two matching pulse patterns will be selected, allowing potentially more nuance to be captured compared to ACELP's single pattern. Codecs such as G.729 implement CS-ACELP techniques in their encoding [23].

2.2.5 SILK: Jointly Optimising LPC and LTP

The Opus codec comprises two sub-codecs: SILK and CELT [36][38]. During encoding, Opus decides between SILK and CELT based on the input's characteristics. Typically, SILK, which is optimised for speech, is used for lower bit rates and more tonal sounds. On the other hand, CELT, designed for generic audio, is applied for higher bit rates

and sounds with a wider frequency spectrum. In this subsection, we will focus on understanding SILK, which is the predominant sub-codec used for speech signals.

SILK offers a unique approach in how it conducts LPC and LTP. In contrast to traditional methods where LPC precedes LTP (such as in GSM), SILK conducts LTP ahead of LPC with the aim of performing pitch estimation before it moves on to spectral envelope modelling. This choice of order implies that details in the representation of pitch is first removed, inadvertently causing the loss of certain spectral details related to pitch harmonics, which could compromise the subsequent LPC analysis. However, SILK employs a feedback mechanism that enables joint optimisation of the LTP and LPC operations. After the initial LPC analysis, the resulting spectral model informs and refines the LTP coefficients, using the original signal as a reference. Similarly, the updated LTP result, now bearing a closer resemblance to the original signal, provides a more accurate foundation for the ensuing LPC stage, improving the overall spectral envelope estimation.

This joint optimisation strikes a strong balance between preserving spectral and pitch details, respectively attributed to the LPC and LTP operations. Despite introducing more computational complexity due to this iterative process, SILK (and Opus) is able to produce efficient and high quality voice encodings.

2.2.6 Codec 2: Harmonic Sinusoidal Coding (HSC)

Novel encoding methods can pave the way for more efficient compression, which is especially relevant in challenging communication environments. Recognising that voiced signals prominently display strong harmonics of the fundamental frequency or pitch, Codec 2 utilises Harmonic Sinusoidal Coding (HSC) [14] to focus on encoding these harmonics. In HSC, the signal is represented almost exclusively as a combination of harmonic sinusoids with varying amplitudes. Unlike most codecs that encode parameters for both the spectrum and the excitation signal, Codec 2 primarily encodes a single set of parameters: the amplitudes of these harmonic sinusoids. As a result, its data stream is able to achieve extremely low bit rates.

The fundamental frequency of the signal is first determined by analysing the input signal. The frequencies of the HSC sinusoids can then be derived by simply working out multiples of the fundamental frequency.

The next step involves sorting out the corresponding amplitudes of each of the HSC sinusoids. This is done through LPC, which is effective in capturing the spectral

envelope. Although the spectral envelope isn't directly encoded, it shapes the harmonic structures by modulating the amplitudes of all component frequencies within, and therefore can be used to infer the required amplitudes.

Having determined both the fundamental frequency and the harmonic amplitudes, the essential parameters required are sorted and can be encoded. This approach focuses on the core structures of voiced signals, ensuring that only the most salient features are preserved while drastically reducing the data rate. As a result, Codec 2 stands out for its efficiency and ability to operate in scenarios demanding ultra-low bit rate transmissions.

2.3 Perceptual Compression

Perceptual codecs operate based on the understanding that uncompressed audio carries a significant amount of information that human cognitive functions do not process. This concept is deeply rooted in the study of *psychoacoustics*, which explores how humans perceive sound, considering the interactions between the ear, the brain, and their inherent limitations.

Research has demonstrated that human hearing can be partitioned into roughly 24 critical bands [7][27]. The locations and spacing of these frequency bands adhere to the logarithmic relationships outlined by psychoacoustics. Dominant sounds within these bands introduce a masking effect, essentially hiding weaker sounds within the same critical band. This phenomenon, known as simultaneous auditory masking or simply simultaneous masking, takes place in the frequency domain [9][15].

Notably, masking can also occur in the temporal domain, where dominant sounds assert a similar masking effect on weaker sounds occurring within a short time before and after them. The duration of the masking is dependent on the strength and length of the dominant sound, but typically lasts about 50ms for pre-masked sounds and up to 300ms for post-masked ones.

Psychoacoustic phenomena indicate that certain information can be omitted without significantly affecting human perception of the sound. When encoding an audio sample, the most dominant sounds can be used to determine which quieter sounds—both in terms of frequency and timing—are effectively masked by them. These masked sounds can be removed, resulting in reduced data sizes, while maintaining audio quality to the human ear.

Perceptual audio codecs, such as MPEG-1 Audio Layer 3 (MP3), and Advanced Audio Coding (AAC) leverage these principles. They discard data following this logic

to save on storage space. For instance, these methods minimise signal amplitude redundancy by employing fewer quantisation steps than a human can discern. Together with a subsequent lossless compression step (such as Huffman coding), perceptual codecs like MP3 and AAC are able to significantly reduce the size of audio files while maintaining a high perceived audio quality.

Chapter 3

Implementation

This chapter discusses how our experiments were set up, their scope, and our considerations. We start by reviewing our selection and scoping process for the speech dataset, list of audio codecs for testing, and choice of ASR systems. The experimental procedure is then explained. Finally, we examine some of the limitations and gaps of the experimental scope.

3.1 Speech Dataset

We reviewed several speech corpora including LibriSpeech [17] and Noizeus [10], before settling on the corpus developed and referenced by [33] and [34]. This corpus, available at [32], was deemed suitable because it contained a large amount of speech data. This ensured that our results came from a large enough population and were sufficiently robust.

Secondly, the speech data was also recorded at a high original resolution of 48kHz. This gave us more flexibility to experiment with altering the data's quality through different combinations of downsampling and compression.

Thirdly, the corpus included transcripts for all the speech samples. This was advantageous for us because these could be used as the reference transcripts in the calculation of WER.

Finally, the corpus had two parallel sets of data. The first set was sourced directly from the original Voice Bank Corpus developed by the Centre for Speech Technology Research at the University of Edinburgh [39], and represents the unadulterated 'quiet' speech set. The second set in the corpus, referred to as the 'noisy' speech set, was derived from the original samples by incorporating various types of noise at varying

Signal-to-Noise Ratio (SNR) levels of 0dB to 15dB. The noises include environmental sounds, mechanical disturbances, and background conversations, among others. This range of noise conditions was selected to simulate a variety of real world scenarios. With both sets, we are thus able to measure the effects of audio compression in quiet and simulated noisy environments.

To maintain a manageable total submission size, we further streamlined the speaker population in the dataset, reducing the count from 56 to just 6 speakers. The selected group of six speakers consisted of a gender-balanced pair from each of the following regions: America, Ireland, and Scotland. Despite the reduction in speaker population, our 6-speaker dataset provided over 105 minutes of speech—a duration deemed reasonable and substantial enough for our subsequent experimentation.

3.2 Audio Compression

We identified 9 lossy audio codecs to investigate their individual and comparative performances. They are: G.726, GSM (Fullrate), Speex, AMR, G.729, Opus, Codec 2, MP3, and AAC, and their techniques have been reviewed in Chapter 2.

These codecs were also selected because they supported the compression of narrowband audio (audio sampled at 8kHz) to low bit rate (32kbps and below) streams, which is the focus of this study. By constraining and controlling our scope to this specific range of audio settings, we are able to provide sharper comparisons of performances based solely on the codec designs. This focus is particularly relevant for applications where bandwidth is limited and low bit rates are necessary, a common scenario in our world with an ever-increasing demand for efficient use of bandwidth.

Compression using G.726, GSM, Speex, AMR, Opus, and MP3 was performed using the `ffmpeg` tool [3][8] running on a Windows operating system. The corresponding libraries required for each codec are shown in Table 3.1. `ffmpeg` was used to downsample the original audio from 48kHz to 8kHz and perform codec encoding in a single command.

In the case of Opus, two *application types* were specified to the encoder, yielding two different configurations. The first was the default `audio` configuration, which Opus recommends to use when the application requires a faithful reproduction of the original input. In this configuration, we forced the encoder to operate in constant bit rate (CBR) mode, as it was observed to deviate from our specified bit rates when the default variable bit rate (VBR) mode was used. The second setting was the `voip` configuration,

Table 3.1: Software encoders, libraries and bit rates used in our experiments

Codec	**Encoder**	**Library**	**Bit rates (kbps)**
G.726	`ffmpeg`	`libavcodec`	16, 24, 32
GSM	`ffmpeg`	`libavcodec`	13
Speex	`ffmpeg`	`libspeex`	2.15, 3.95, 5.95, 8, 11, 15, 18.2, 24.6
AMR	`ffmpeg`	`libopencore-amrnb`	4.75, 5.15, 5.9, 6.7, 7.4, 7.95, 10.2, 12.2
G.729	Asterisk	-	8
Opus	`ffmpeg`	`libopus`	6, 8, 10, 12, 16, 24, 32
Codec 2	`c2enc` `c2dec`	-	0.45, 0.7, 1.2, 1.6, 2.4, 3.2
MP3	`ffmpeg`	`libmp3lame`	8, 16, 24, 32
AAC	`qaac`	-	8, 12, 16, 20, 24

which was recommended for applications demanding higher levels of intelligibility. Presumably this would influence the decision process to favour the SILK sub-codec rather than CELT. The default VBR mode was used here, and the encoder was found to adhere to our specified bit rates.

For the AAC codec, we used the `qaac` command line utility [2] running in a Windows environment. Like `ffmpeg`, `qaac` was able to downsample and encode the audio at the same time.

To encode with G.729 and Codec 2, we used the Asterisk tool [1] and utilties available at [20] respectively. These tools were operated within a Linux environment. In both cases, the `sox` tool was used to downsample the files to 8khz prior to compression.

The primary goal of the experiments was to fully understand the behaviors and performances of these codecs under different compression conditions, particularly at lower bit rates. To this end, speech was compressed at varying bit rates below 32kbps. These bit rates were chosen based on the options the codecs supported, and for us to achieve a reasonable spread of data collection. These details and the specific ranges used for each codec in our experiments are summarised in Table 3.1.

3.3 ASR Services

We used ASR services offered by Rev AI [5] and Google Cloud Platform [6] to obtain the transcripts of our submitted samples. Having two ASR systems allowed us to establish a system of cross-validation where the outputs of one service could corroborate the findings of the other. This would enhance the robustness and interpretability of our results, strengthening their validity and mitigating the risk of reliance on a single ASR service.

Both ASR services were assessed to be ideal for our purposes. At the time of writing, they were both reputable players in the field, producing transcriptions with market-leading accuracy. Their high accuracies, considered state-of-the-art within the industry, ensured that our experiments could be focused on the effects of compression, and not be bottlenecked by transcription ineffectiveness. In addition, they offered reliable services with high uptime and availability. All of these characteristics made them suitable for our experiments, ensuring that large amounts of data could be processed reliably and expediently.

Both services were accessed using their exposed Python Application Programming Interfaces (APIs). English was specified as the language, and we opted to disable speaker diarisation (distinguishing between different speakers in a recording) in the transcripts because only a single intended speaker was present in the recordings.

Rev AI rejected the transcription of audio that were shorter than two seconds. For this reason, we only submitted samples longer than that threshold to both ASR services. Altogether, each submitted batch of files comprised over 105 minutes of audio data.

3.4 Intelligibility and Word Error Rates

Given a generated transcript and a reference transcript, the word error rate (WER) can be derived using the equation $WER = \frac{S+D+I}{N}$, where S, D, and I are the number of substitution, deletion and insertion errors found in the generated transcript, and N is the total number of words in the reference. For example, if the reference of $N = 6$ is “a cat sat on the mat” while the generated transcript reads ”her cat on mat hat”, it will be determined that $S = 1$ (“a”/“her”), $D = 2$ (“sat”, “the”), and $I = 1$ (“hat”). Further, this results in a $WER = .66$.

WER provide a straightforward metric for assessing the intelligibility of ASR output. One notable limitation of using WER, however, is that it does not differentiate

between major and minor errors. For instance, a transcription of "it's" will register two errors ($WER = 1.0$) if the reference is "it is", despite these two phrases having identical meanings. A more sophisticated metric could potentially account for the varying severities of errors, yielding a more nuanced intelligibility score. Nevertheless, we chose to use WER due to its ease of implementation and its widespread acceptance as a measure of ASR output. Importantly, our study aims to investigate the impact of compression on speech intelligibility, which requires relative rather than absolute measures. Therefore, we are primarily utilising WER as a relative metric, to compare the performance of different ASR outputs under various levels of compression, rather than treating it as an absolute measure of intelligibility.

Generated transcripts from the ASR platforms and reference transcripts were first normalised by removing special characters and lowercasting, so that subsequent textual comparison would be streamlined. The WER can then be determined by considering the *Levenshtein distances* [13] or edit distances between the words across both strings. This calculation was performed using an adapted Python script from [40]. In order to generate intelligibility scores for entire batches, overall WER were then calculated by aggregating the word errors and word counts from each sample file's transcripts.

3.5 Experimental Procedure

From the master corpora [36], we retrieved the entire set of speech samples from the 6 identified speakers, alongside their transcripts. Both the `clean` and `noisy` sets were obtained, corresponding to speech in quiet and simulated noisy environments respectively. The 6 speakers contributed a total of 2301 files, which were recordings of speech ranging from short phrases to entire sentences. We will henceforth refer to each set of 2301 files as a *batch*.

The batches of speech are then compressed using the software, libraries, and according to the bit rates outlined in Table 3.1. Separate batches of speech were resampled at 8kHz using `sox`, but were not compressed. These batches would be used to establish the baselines for *downsampled, uncompressed* speech.

Python scripts based on Jupyter Notebooks were written to interface with the Google Cloud Platform and Rev AI ASR APIs. Using the API, speech files were successively uploaded to the platforms and the transcribed results returned. Because Rev AI rejected the transcribing of files shorter than 2 seconds in duration, our scripts did not submit these files to either Rev AI or Google Cloud services.

After some text normalisation, the returned transcripts were compared to the original transcripts to find errors. The Python code from [40] was incorporated to detect word substitution, deletion, and insertion errors and thereby determine each file's WER. Finally, the total errors for all the files in a batch were aggregated and the batch's WER calculated. These WER were then recorded, and the process repeated for each other batch with differing codec, bit rate, or noise setting.

3.6 Assumptions and Limitations

In the course of our experiments, several assumptions were made in the face of challenges or limitations in the set up. This section addresses these issues and their potential impact to give the reader a more comprehensive understanding of the circumstances of our findings.

3.6.1 Effects of Varying Noise Conditions Within Noisy Set

As we have mentioned in Section 3.1, different noises were used to layer over the original speech samples and create the noisy set of data. Each speech file underwent different noise conditions, involving a variety of sounds—from environmental noises to mechanical disturbances and background conversations—with SNR levels ranging from 0dB to 15dB. Given this non-uniform application of noise, it is plausible that certain noise conditions would affect ASR transcription accuracy and WER more than others.

While the intricate landscape of noise variation holds potential implications for interpreting the ASR performances, our study did not probe into the detailed influence of individual noise types or SNR levels on the WER. Nevertheless, we surmise that these variations are unlikely to significantly skew our overall findings. WER were aggregated over a large numbers of diverse samples, dispersing the impact that any individual noise condition might have on the overarching trends.

Furthermore, the focus was to study the degradation of speech quality across varying codecs and bit rates. In this context, we note that while SNR levels and noise types varied from one sample to another, the noise condition applied to each individual sample remained consistent across different compression and bit rate settings. This ensured that any comparisons or trends we observed among the codecs or bit rates was unaffected by varying noise characteristics across the batches.

3.6.2 Effects of Using Alternative Software or Libraries for Compression

While the software and libraries presented in Table 3.1 are some of the recommended tools to use for encoding audio, we note that in some cases there are alternative tools that can be used for the same codec. For example, although we used `ffmpeg` with the `libopus` library to encode with Opus, an alternate encoder `opusenc` is offered by the Opus project at [4]. Such variants of tools and their versions are likely to have differences in their algorithmic implementations at the code level, thereby introducing subtle differences in encoding quality.

Considering every encoding tool and its multiple versions While recognising these differences from varied algorithmic implementations, we deem these differences to be negligible. The encoding tools used nevertheless align with the published standards of the codecs, making them representative of the codecs' intended performance. Furthermore, given our emphasis on broader trends and overarching patterns, we believe that any minor discrepancies will be eclipsed, diminishing their influence on our final conclusions.

3.6.3 Effects of Additional Conversions to Linear PCM

The online ASR systems were found to only accept audio files in certain formats and extensions. For example, Google Cloud Platform rejected the `.g729` and `.mp4` formats corresponding to files encoded using the G.729 and AAC codecs respectively. On the other hand, certain formats like `.amr` were directly accepted by the platform. For these formats that were not directly accepted, we converted the audio files into 16-bit linear PCM files inside `.wav` containers before submission. LPCM was specifically chosen because it is a lossless format, ensuring minimal data degradation during this additional step.

We recognise that converting audio to 16-bit LPCM might nevertheless introduced unwanted effects such as added quantisation noise. This is an inevitable byproduct of any encoding process itself. In theory, this could lead to slightly skewed comparisons, especially when set against batches like the `.amr` files that did not require this extra step. That said, a bit depth of 16 bits was specifically chosen to minimise the impact of such quantisation noise. Moreover, any subtle degradation effects arising from this additional step are likely to be overshadowed by the more pronounced effects inherent to codec compression, which this study is focused on. Consequently, this study assumes

that the impact of this conversion on the ASR results is insignificant.

3.6.4 Periodic Updates or Changes to ASR Services

ASR systems are dynamic and undergo modifications over time. These modifications can result from improved algorithms, better training data, platform upgrades, or evolving speech patterns. While intended to enhance accuracy and applicability, these updates can subtly alter the system's behavior and its transcription results.

During our study, we treated the ASR systems' consistency as an assumption rather than a certainty. Silent updates to the platforms might have occured without our knowledge. To counteract the potential variability introduced by such changes, we condensed our experiments into a short timeframe, reducing the risk of significant system changes during our study.

Given the variable nature of ASR systems, outcomes such as ASR transcripts and WER might not be entirely reproducible in future iterations of our experiments. Yet, the overarching trends observed in our research should hold in similar studies. Since any modifications to the ASR systems would uniformly impact all submissions, a diverse dataset should ensure that the primary trends persist. In summary, while exact outcomes from ASR systems might fluctuate, the core insights from our research are transferable to future studies.

Chapter 4

Results

This chapter summarises the results from our experiments detailed earlier. We will first review the collected WER data from the various tested scenarios. Then, we will examine the results and provide commentary on our observations in the following chapter.

4.1 Speech in a Quiet Environment

The WER of the transcribed speech files, downsampled to 8kHz and compressed using the various codecs, are plotted across the varying bit rates specified as shown in Figures 4.1 and 4.2 for the Google Cloud and Rev AI ASR platforms respectively. Along with the plotted data, trendlines have been drawn to illustrate the correlations between the bit rates and their corresponding error rates. Baseline WER for the original speech (Uncompressed 48kHz), and downsampled but uncompressed speech (Uncompressed 8Khz) are shown for comparative purposes. The detailed results for each submitted batch can be found in Appendix A.1.

WER was observed to consistently worsen as bit rates were reduced, a trend in line with our experimental expectations as reducing bit rates results in decreasing the amount of information available and should lead to higher inaccuracies. This observation held true when considering variations within a single codec, and also as a broader trend across all codecs. Additionally, the degradation of WER was more pronounced at lower bit rates, where even slight reductions led to substantial increases in error rates. On the other hand, improvements to WER at the higher bit rates were less noticeable.

With both ASR systems, the effect of downsampling of the speech samples from 48kHz to 8kHz is seen to have a slight but measurable impact to WER. Irrespective of codec choices or the encoding bit rates, WER were not seen to improve past the

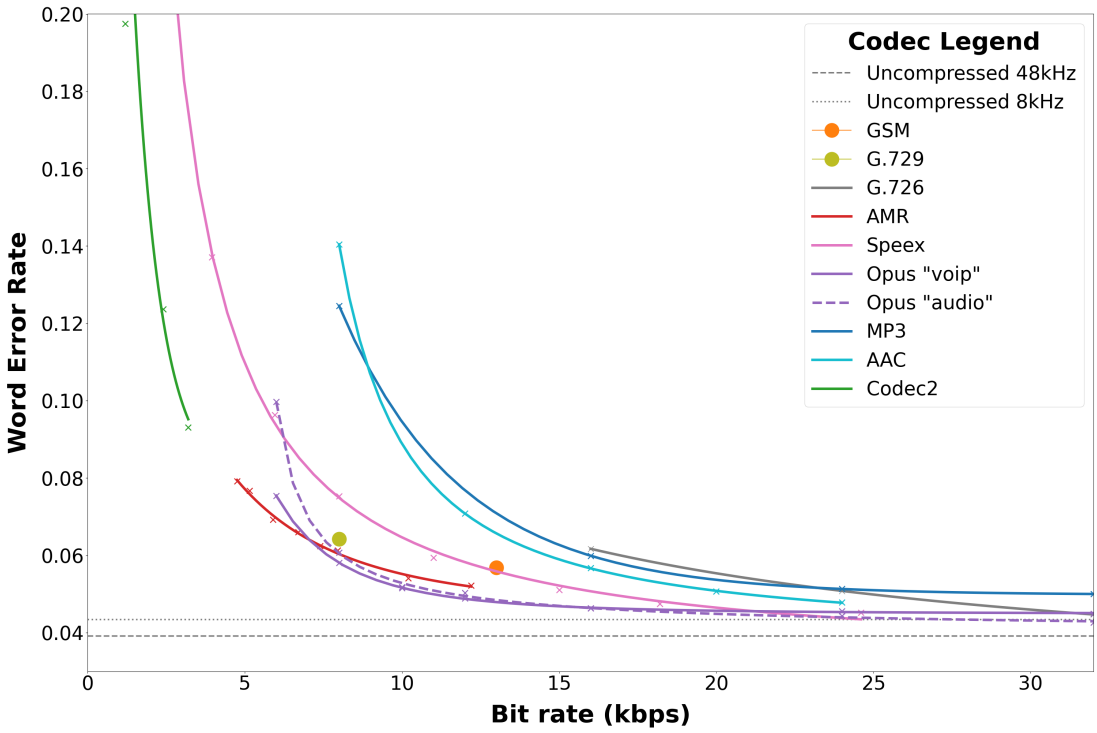

Figure 4.1: WER of speech samples in a quiet environment, downsampled to 8kHz, using different codecs at varying bit rates, and transcribed using Google Cloud ASR. WER for uncompressed speech, at 48kHz and 8kHz resolutions are also shown.

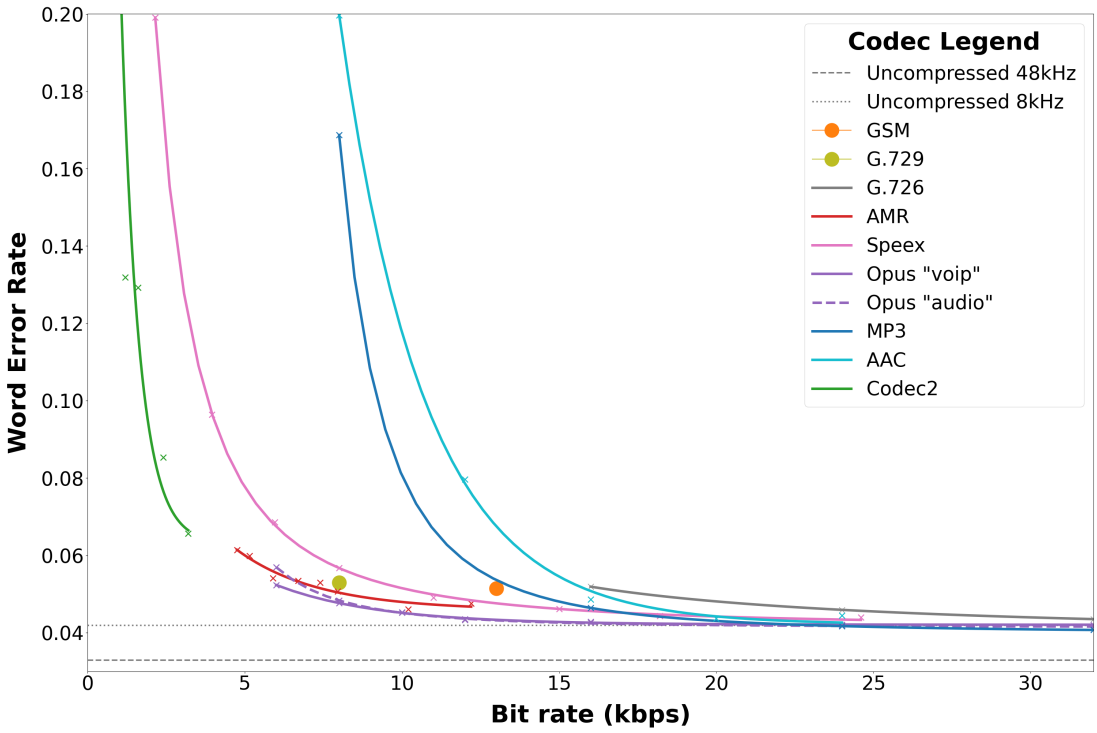

Figure 4.2: WER of speech samples in a quiet environment, downsampled to 8kHz, using different codecs at varying bit rates, transcribed using Rev AI ASR. WER for uncompressed speech, at 48kHz and 8kHz resolutions are also shown.

baseline of uncompressed, but downsampled speech.

Without the introduction of noise, error rates were also observed to be maintained at reasonable levels despite the effects of compression. WER were generally below 0.2 for bit rates above 1kbps. While not an absolute judgement of accuracy by itself, it is notable that modern ASR systems are able to transcribe speech that has undergone substantial levels of compression with relative accuracy.

4.2 Speech in Simulated Noisy Environments

Similar to the previous section, we plotted the WER of the transcribed speech in simulated noisy environments, downsampled to 8kHz and compressed at varying bit rates using the different codecs codecs. Corresponding plots to Google Cloud's and Rev AI's ASR services are shown in Figures 4.3 and 4.4 respectively, with trendlines further demonstrating the correlations between the bit rates and transcription accuracy. Baseline WER for uncompressed speech, at both the full 48kHz resolution and downsampled 8kHz, are also shown. The detailed results for each submitted batch can be found in Appendix A.2.

WER across the board were noticeably worse than before, consistent with our expectation that additive noise would have an adverse impact on ASR transcription accuracy. Baseline WER of uncompressed speech at the full 48kHz resolution had at least doubled from less than 0.05 in the quiet environment, to 0.10 or more in the simulated noisy environments. At low bit rates, it was common for WER to exceed values of 0.25, demonstrating the severe effects of added noise.

As with the previous results of speech in a quiet environment, WER was observed to consistently worsen with reducing bit rates. The trend was similarly observed with varying bit rates within the same codec, and also as a broader observation across all codecs. Effects of worsening WER were also sharpened at lower bit rates, and less pronounced at high bit rates.

An outlier was observed when speech compressed with the AAC codec and transcribed by Google Cloud ASR. A WER of about 0.23 was measured when the speech was compressed at a bit rate of 12kbps. This result significantly outperformed an expected WER of 0.28, and even surpassed the WER of about 0.26 at 16kbps.

Interestingly, compression sometimes improved WER over the uncompressed baseline. Uncompressed, downsampled speech posted a WER of about 0.21 on Google Cloud ASR. Yet, WER were seen to *improve* when compression was further performed.

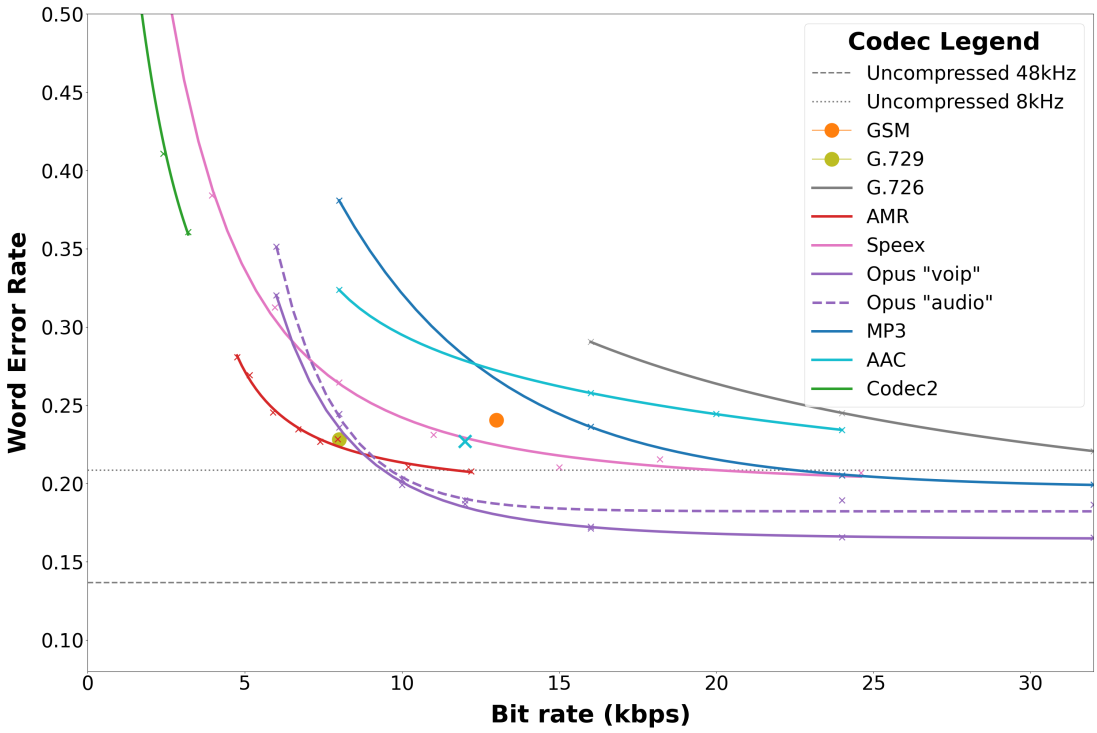

Figure 4.3: WER of speech samples in simulated noisy environments, downsampled to 8kHz, using different codecs at varying bit rates, and transcribed using Google Cloud ASR. WER for uncompressed speech at 48kHz and 8kHz resolutions are also shown.

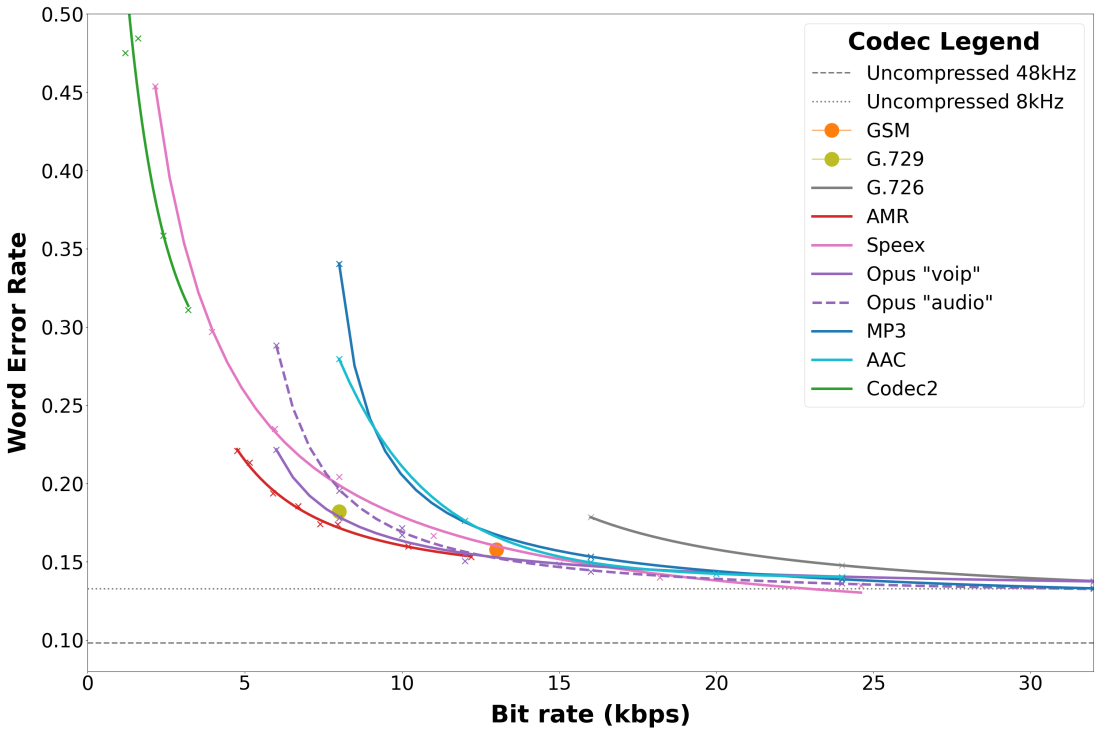

Figure 4.4: WER of speech samples in a simulated noisy environments, downsampled to 8kHz, using different codecs at varying bit rates, transcribed using Rev AI ASR. WER for uncompressed speech, at 48kHz and 8kHz resolutions are also shown.

This effect was most noticeable with the Opus codec at bit rates above 10kbps, and to a lesser degree with Speex and MP3 at bit rates above 24kbps. This behaviour was not seen with speech from the quiet environment, nor was it observed with submissions to Rev AI's ASR platform. Possible reasons for this will be discussed in Subsection 5.2.

Chapter 5

Discussions

In the sections that follow, we reflect on several notable observations from Chapter 4. Specifically, we delve into the factors that influence variations in codec performance, the unexpected finding that compression enhanced WER, and propose our experimental approach as an alternative method to evaluate codec efficacy.

5.1 Relative Performances of Codecs

The trendlines in Figures 4.1 through 4.4 suggest that certain codecs perform better than others. For example, the Opus codec is seen to impact transcription accuracy less than Speex, based on the relative positions and inclinations of their curves. The former's WER and degradation of WER are overall less severe than the latter's across the range of bit rates tested. Similarly, the relative positions of MP3 and G.726 suggest that the former consistently performs better than the latter, as far as transcription accuracy is concerned.

Table 5.1: Banding of codecs by performance

Band	**Codecs**
I	AMR, G.729, Opus
II	GSM, Speex
III	AAC, MP3
IV	G.726

To aid further discussion, we therefore propose the above banding of codecs in Table 5.1 based on their performances. Codecs belonging to a higher band are judged to generally have better WER than those of a lower band, while codecs within the same

band have similar WER that cannot be meaningfully differentiated based on the current data alone.

Codec 2 stands out as a specialised codec operating at very low bit rate of 3.2kbps and below. It was left out of the banding exercise because there was a lack of comparison with other codecs operating at the same range of bit rates (0.45 to 3.2 kbps). We note, however, that Codec 2 appeared to perform better than Speex between 2.15kbps and 3.95kbps.

5.1.1 Encoding Efficiencies Between Bands

We examined the performance differences in WER between the proposed bands and formulate potential explanations, starting with the lower bands:

5.1.1.1 Band III (AAC, MP3) and Band IV (G.726)

The G.726 codec, utilised in Band IV, employed Differential Pulse-code Modulation as detailed in Section 2.1.3. This approach sets G.726 apart from the other codecs we examined, as it does not take advantage of parametric compression techniques. This limitation hindered G.726 from reaching the encoding efficiency levels achieved by its counterparts. As a consequence of this poorer efficiency, the audio quality was more susceptible to degradation, leading to an increased WER.

In contrast, codecs in Band III (and those in the higher bands) make a pivotal shift by incorporating elements of parametric compression, transcending mere waveform compression. The shift in techniques allowed these codecs to characterise more features of a sound signal using the same number of bits. By preserving more of the original signal's integrity in this manner, the encoding efficiency is notably enhanced. This contrast between G.726 and its parametric counterparts illustrates the profound impact that different paradigms of compression techniques can have on the resultant quality and intelligibility of the audio.

5.1.1.2 Band II (GSM, Speex) and Band III (AAC, MP3)

Audio codecs like MP3 and AAC are designed for a wide range of applications and often use perceptual coding (see Section 2.3) that considers the entire range of human hearing, typically from 20 Hz to 20 kHz. While this makes them effective for various types of audio such as music, it might cause them to handle unnecessary information

when it comes to speech, such as frequencies that lie in the higher and lower extremes of the above range.

Human speech primarily occupies a frequency range of 300 Hz to 3.4 kHz [21], whereas general-purpose audio codecs like MP3 and AAC handle a broader spectrum. These codecs are potentially less optimised at handling speech-specific properties such as formants and coarticulation. This may then lead to relative inefficiencies in compression when the target audio is primarily speech.

In contrast, speech-specific codecs in Band II (and Band I) like GSM and Speex are crafted to focus on this narrower band and the specific characteristics of speech. By concentrating on the relevant frequencies and speech phenomena, less critical information, such as non-speechband frequencies, consume less of the allocated bit depth. The result is a more efficient compression for speech, given the same bit rate.

5.1.1.3 Band I (AMR, G.729, Opus) and Band II (GSM, Speex)

The codecs listed in both bands are all tailored towards speech applications and use the various parametric encoding techniques reviewed in Section 2.2. While they all borrow from the same overarching principles of the source-filter model, the specific techniques that they employ have varying degrees of sophistication, thereby contributing to different levels of efficiency and performance.

GSM and Speex are dependent on the RPE-LTP and CELP techniques (see Subsections 2.2.4.1 and 2.2.4.2) respectively. A major downside of RPE-LTP is that it oversimplifies the complex characteristics of human speech by attempting to represent the speech signal with a regularised pulse model. By design, the number of pulses encoded is regulated, which limits the repertoire of possible excitation signals. On the other hand, CELP's final encoding stage relies on a fixed, predefined codebook, which also limits its range of possible candidate signals. In both cases, the limited range of excitation signals often lead to mismatches and quantisation errors between the original speech signal and the encoded approximations.

In contrast, the ACELP and CS-ACELP techniques that AMR and G729 employ (see Subsections 2.2.4.3 and 2.2.4.4) are more sophisticated and cater to a wider range of excitation signals. Rather than constraining themselves to a finite codebook, these techniques use algebraic representations that can flexibly adapt to many unanticipated signals and variations. This significantly improves the accuracy of the encoded approximations by reducing signal mismatches and quantisation errors. Consequently, this leads to fewer decoding artifacts and superior performance in terms of speech

intelligibility.

Opus' SILK subcodec is also able to enrich its repertoire of excitation signals because of its joint optimising of LPC and LTP (see Subsection 2.2.5). The concurrent evaluation of the spectral shape and periodicity of the signal, through LPC and LTP respectively, enables SILK to generate a varied, nuanced, and adaptive array of excitation signals. Although this approach is computationally more demanding, speech synthesised through SILK bears a closer resemblance to the original, with more natural variations and fewer artifacts.

In summary, the differences in performance can be attributed to the encoding techniques employed, which determine the extent of variability that excitation signals can exhibit. Band II's techniques of RPE-LTP and CELP limit themselves to a relatively narrow range of excitation signals and are less effective at capturing the full nuances and variability of speech. Conversely, ACELP, CS-ACELP, and SILK's joint optimising afford a wider range of signal adaptability, enabling a more dynamic representation of speech patterns. This allows for the Band I codecs to provide richer and more intelligible speech.

5.1.2 Summary

The relative performances of codecs across different bands highlight the significant impact that encoding designs and techniques can have on transcription accuracy. While adjustments to bit rates provide opportunities for performance tweaking, the encoding strategies rooted in the principles of audio compression and psychoacoustics ultimately dictate the broader success of the resultant speech intelligibility and transcription accuracy.

5.2 Compression Leading to Improved WER

We observed that compression using Opus, Speex or MP3 resulted in an improvement to WER, when a traditional understanding of audio fidelity suggests that WER should have worsened instead. This unexpected deviation can be attributed to noise suppression, clarity enhancement and the perceptual coding techniques found in these codecs.

5.2.1 Noise Suppression and Clarity Enhancement

Besides audio compression, both the Speex and Opus codecs employ techniques to mitigate noise and enhance voice clarity. Speex integrates a noise reduction feature in its preprocessor module to ascertain and mitigate noise [35]. By analysing the speech frames, differentiated gains are applied to each audio segment with the aim of optimising and reducing the overall signal-to-noise ratio (SNR). Optimising the SNR in this way effectively reduces the noise components and allows the speech components to stand out against background disturbances.

Similarly, Opus offers its own suite of noise suppression tools within the codec [38]. It establishes background noise levels by evaluating the averaged energies across time frames. This method effectively computes the extent of background noise in the audio and forms a distinction between the primary voice signal and other sources of disturbances. By doing so, it is then able to isolate and suppress the background noise. Secondly, Opus also employs an adaptive high-pass filter, which dynamically adjusts its cutoff frequency based on the detected pitch period of the signal. In this way, the filter systematically removes peripheral low frequency components that could potentially obscure the speech signal. This combination of tools work in concert to enhance speech clarity and definition.

Although lossy compression compromises on the audio fidelity itself, the enhancements to speech clarity and definition can more than compensate for the degradation. This would result in an improvement to the WER that surpasses the baseline of uncompressed speech.

5.2.2 Enhancements Through Perceptual Coding

Perceptual coding and compression, which we reviewed in Section 2.3, can also have an ancillary effect of enhancing speech signals. While the primary purpose of perceptual coding is to discard data and reduce transmission sizes, it is possible that noise is discarded in the process.

Recall that perceptual coding capitalises on psychoacoustics, which include the human auditory system's tendency to mask weaker sounds when faced with more dominant counterparts, both in the temporal and frequency domain. Codecs like MP3 therefore discern and selectively discard these softer, 'masked' sounds, retaining only the dominant audio components that resonate most with human ears.

Although MP3 was primarily designed for music, these psychoacoustical principles

can have beneficial outcomes in speech applications. For example, if there is ambient noise or background interference that is being overshadowed by the primary speech sounds, perceptual coding's removal of 'masked' sounds could effectively pare down these disturbances, yielding a clearer speech signal.

However, this very process of discarding information might inadvertently remove subtle yet crucial speech components such as softer consonants. The loss of such elements can prove detrimental to speech intelligibility, affecting the WER. The potential for improving signal clarity by discarding noise is countered by the possibility of losing intelligibility due to discarding speech information, and might explain why MP3's WER gains, while evident, do not match the gains seen with Opus.

5.2.3 Differences Between ASR Platforms

The above phenomenon was observed specifically on the Google Cloud Platform (GCP), but not with Rev AI's system. While we cannot ascertain the reasons without a thorough examination of their proprietary systems, our hypothesis is that variations in the way each platform preprocesses the submitted speech might account for the discrepancy.

We speculate that during preprocessing, Rev AI incorporated more substantial noise suppression compared to GCP. This notion is further supported by our observations that Rev AI consistently achieved lower WER than GCP across all tests with simulated background noise. Furthermore, as depicted in the WER of speech in simulated noisy environments (Figures 4.3 and 4.4), we see that the curves corresponding to various codecs tend to converge at the baseline value of around 0.13 as bit rates increase. This pattern contrasts with GCP, where the curves display a more pronounced dispersion. These observations suggest a likelihood that Rev AI has implemented preprocessing techniques that consistently impacted all codecs in a uniform manner, leading them to converge around similar performance outcomes. In GCP's case, such preprocessing might not be as effective, or was entirely absent, which would account for its more dispersed pattern of curves.

Rev AI's suppression likely surpassed those of Opus, Speex, and MP3 covered previously, as evidenced by the similar error rates and their inability to surpass the uncompressed baseline WER of 0.13. Irrespective of the individual capabilities of each codec, the preprocessing by Rev AI provided consistent and superior noise management across the board. This distinct advantage in noise management, not observed in GCP, offers a plausible explanation for the unique phenomenon.

5.2.4 Summary

In summary, the paradoxical observation that audio compression can lead to improved WER underscores the intricate balance between signal clarity and information preservation. While compression codecs may degrade audio fidelity, their integrated noise suppression and enhancement tools can improve speech clarity. However, the outcomes can vary based on the specific ASR platform and its preprocessing capabilities. It is crucial to consider both the inherent capabilities of codecs and the preprocessing techniques of ASR platforms when evaluating the performances of codecs in speech recognition tasks.

5.3 Benchmarking Speech Quality Using ASR WER

The endeavor to assess and rate the quality of speech is an ongoing field of study. Diverse methods, both subjective and objective, have been employed to this end, each presenting its unique set of challenges.

Subjective measures are a common way to conduct evaluation of the quality of speech. They are an obvious choice because they directly tap into the human experience, capturing the intricacies and nuances of our auditory percepts. Methods such as using *Mean Opinion Scores* (MOS) [24] or the *MUltiple Stimuli with Hidden Reference and Anchor* (MUSHRA) [31] have become staples for these sorts of evaluation. These methods involve subjects rating speech samples on a numerical scale, with and without reference samples, respectively. While the results of subjective measures are straightforward to interpret, they naturally introduce variability and inconsistency. Factors such as listener fatigue, cultural biases, and environmental variability can potentially skew results in unpredictable ways [19].

On the other hand, objective measures seek to reduce such variabilities in the results. The *Perceptual Evaluation of Speech Quality* (PESQ) [25] and *Perceptual Objective Listening Quality Assessment* (POLQA) [26] instruments are two prime examples of this category. These methods compare a degraded speech sample against an original reference, and by simulating human auditory processes such as auditory masking, produce scores that reflect the extent of degradation. Despite their sophistication, these tools sometimes grapple with mirroring the intricacies of human auditory processing. The auditory experience is riddled with numerous physiological and psychological factors that researchers are yet to fully understand, posing limitations on the robustness of such

measures. Furthermore, the ratings produced by PESQ and POLQA are relational to the reference sample, meaning that any inherent issues with the reference can affect the final score, potentially leading to misrepresentations of the true quality of the test sample.

We propose for the use of ASR systems either as an alternate, or supplementary technique to the above means. Leveraging WER as an evaluative metric, ASR systems present a level of consistency that is often challenging to maintain in human-centric evaluations. Unlike human listeners whose judgements can be affected by factors such as fatigue, biases, or external distractions, ASR evaluations remain stable across iterations. The straightforward nature of WER also provides an objective and empirical lens to speech quality in terms of its intelligibility and clarity. If an ASR system produces a high WER, it could suggest potential comprehension issues mirrored in human listeners. In this light, incorporating ASR-derived metrics like WER with established subjective and objective measures promises a richer, multi-dimensional perspective on speech quality assessment.

Chapter 6

Conclusions

Our paper delved into the realm of lossy audio codec compression and encoding mechanisms, and their interplay with ASR systems. We started with a comprehensive review that traced the evolution of audio compression techniques. The study charted the progression from foundational constructs to the advanced algorithms that dominate today's digital audio landscape. Psychoacoustics further enriched our understanding, illuminating the auditory cognitive experience that codecs can mirror to achieve higher efficacies. Concurrently, we introduced the paradigms of parametric compression against waveform compression, and explained how the former leveraged the peculiarities of speech to achieve superior performance over the latter.

We explored these performance differences by compressing speech samples using a multitude of codecs at low bit rate settings, where the differences were expected to be more pronounced. Instead of the currently established means of evaluating speech quality using subjective tests or objective algorithms like the PESQ and PQLOA, we opted to use online ASR platforms, taking advantage of their consistency and availability. Central to our approach was the adoption of WER as a robust and empirical measure of performance, enhancing our ability to conduct nuanced comparisons and analyses of codec performances.

Our data and comparative analyses echoed the anticipated trend that codec performances directly corresponded to their sophistication of encoding techniques. This was evident not only in terms of average WER across the tested bit rates, but also in the rate of WER degradation as bit rates were reduced. These disparities underscored the resilience of the more advanced codecs in preserving speech intelligibility under constraints, whilst their less sophisticated counterparts faltered more noticeably.

Through our evaluations, we also discovered that certain codecs can enhance speech

quality and intelligibility even though information was discarded during the process. This was most evident in scenarios with background noise, where the codecs acted as noise suppressors, focusing on and preserving the most salient parts of the audio.

The granularity of our findings validates not only the theoretical distinctions between encoding techniques, but also the effectiveness of our methodology of using ASR systems to evaluate speech quality and intelligibility. We propose this ASR-based approach as a novel method for such assessments. The employment of ASR systems offers a unique blend of consistency, stability, and interpretability that traditional practices might find challenging to uphold. ASR systems therefore emerge as evaluative tools that are indispensable for future research in compression and other speech manipulation techniques.

Bibliography

[1] Asterisk. https://www.asterisk.org/. Accessed: 18 June 2023.

[2] CLI QuickTime AAC/ALAC encoder. https://github.com/nu774/qaac/. Accessed: 19 June 2023.

[3] FFmpeg. https://www.ffmpeg.org/. Accessed: 15 June 2023.

[4] Opus downloads. https://opus-codec.org/downloads/. Accessed: 14 July 2023.

[5] Speech Recognition Service Rev AI. Speech to Text API. https://www.rev.ai/. Accessed: 28 June 2023.

[6] Google Cloud. Speech-to-Text: Automatic Speech Recognition. https://cloud.google.com/speech-to-text/. Accessed: 28 June 2023.

[7] H. Fletcher. Auditory patterns. *Reviews of Modern Physics*, 12(1):47, 1940.

[8] gyan.dev. Codex FFmpeg. https://www.gyan.dev/ffmpeg/. Accessed: 15 June 2023.

[9] S. Hacker. *MP3: The Definitive Guide*. O'Reilly, 2000.

[10] Y. Hu and P. Loizou. Noizeus: A noisy speech corpus for evaluation of speech enhancement algorithms. https://ecs.utdallas.edu/loizou/speech/noizeus/. Accessed: 13 June 2023.

[11] European Telecommunications Standards Institute. Recommendation GSM 06.10: GSM full rate speech transcoding, 1991.

[12] European Telecommunications Standards Institute. Digital cellular telecommunications system (Phase 2+); Adaptive Multi-Rate (AMR) speech transcoding. Technical Report ETSI-EN-301-704 V7.2.1, 2000.

[13] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1965.

[14] R. McAulay and T. Quatieri. Speech analysis/synthesis based on a sinusoidal representation. *IEEE Trans. Acoust., Speech, Signal Process.*, 34(4):744–754, 1986.

[15] B. C. J. Moore. *An Introduction to the Psychology of Hearing*. Brill, 2012.

[16] A. V. Oppenheim, A. S. Willsky, and S. H. Nawab. *Signals and Systems*. Prentice Hall, 2nd edition, 1999.

[17] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur. LibriSpeech: an ASR corpus based on public domain audio books. *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference*, pages 5206–5210, 2015.

[18] M. Ray, M. Chandra, and B. P. Patil. Speech Coding Techniques for VoIP Applications: A Technical Review. *World Applied Sciences Journal*, 33(5):736–743, 2015.

[19] A. W. Rix. Comparison between subjective listening quality and P.862 PESQ score. 2003.

[20] D. G. Rowe. codec2: Open source speech codec designed for communications quality speech between 700 and 3200 bit/s. `https://github.com/drowe67/codec2/`. Accessed: 21 June 2023.

[21] International Telecommunication Union Telecommunication Standardization Sector. Recommendation G.711: Pulse code modulation (PCM) of voice frequencies, 1988.

[22] International Telecommunication Union Telecommunication Standardization Sector. Recommendation G.726: 40, 32, 24, 16 kbit/s Adaptive Differential Pulse Code Modulation (ADPCM), 1990.

[23] International Telecommunication Union Telecommunication Standardization Sector. Recommendation G.729: Coding of Speech at 8 kbit/s Using Conjugate-Structure Algebraic-Code-Excited Linear Prediction (CS-ACELP), 1996.

[24] International Telecommunication Union Telecommunication Standardization Sector. Recommendation P.800: Methods for subjective determination of transmission quality, 1996.

[25] International Telecommunication Union Telecommunication Standardization Sector. Recommendation P.862: Perceptual evaluation of speech quality (PESQ): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs, 2001.

[26] International Telecommunication Union Telecommunication Standardization Sector. Recommendation P.863: Perceptual Objective Listening Quality Analysis (POLQA), 2018.

[27] J. O. Smith and J. S. Abel. Bark and ERB Bilinear Transforms. *IEEE Transactions on Speech and Audio Processing*, 7(6):697–708, 1999.

[28] K. N. Stevens. *Acoustic Phonetics*. Current Studies in Linguistics Series. MIT Press, 2000.

[29] L. Sun, I. Mkwawa, E. Jammeh, and E. Ifeachor. *Guide to Voice and Video over IP: For Fixed and Mobile Networks*. Springer Publishing Company, Incorporated, 2013.

[30] T. E. Tremain. The government standard linear predictive coding algorithm: LPC-10. *Speech Technology Magazine*, pages 40–49, 1982.

[31] International Telecommunication Union. Recommendation BS.1534: Method for the subjective assessment of intermediate quality levels of coding systems, 2001.

[32] C. Valentini-Botinhao. Noisy speech database for training speech enhancement algorithms and TTS models. `https://doi.org/10.7488/ds/2117`, 2017. Accessed: 10 June 2023.

[33] C. Valentini-Botinhao, X. Wang, S. Takaki, and J. Yamagishi. Investigating RNN-based speech enhancement methods for noise-robust text-to-speech. *9th ISCA Workshop on Speech Synthesis Workshop (SSW 9)*, 2016.

[34] C. Valentini-Botinhao, X. Wang, S. Takaki, and J. Yamagishi. Speech Enhancement for a Noise-Robust Text-to-Speech Synthesis System Using Deep Recurrent Neural Networks. In *Proc. Interspeech 2016*, pages 352–356, 2016.

[35] J. Valin. *The Speex Codec Manual*. Xiph.Org Foundation. Accessed: 12 July 2023.

[36] J. Valin, G. Maxwell, T. B. Terriberry, and K. Vos. High-quality, low-delay music coding in the opus codec. *arXiv preprint*, 2016.

[37] J.-M. Valin. Speex: A free codec for free speech. *arXiv preprint*, 2016.

[38] K. Vos, K. Vandborg Sørensen, S. Skak Jensen, and J. Valin. Voice coding with opus. In *Audio Engineering Society Convention 135*. Audio Engineering Society, 2013.

[39] J. Yamagishi, C. Veaux, and K. MacDonald. CSTR VCTK Corpus: English Multi-speaker Corpus for CSTR Voice Cloning Toolkit. `https://datashare.ed.ac.uk/handle/10283/3443/`, 2019. Accessed: 13 June 2023.

[40] Z. Zhao. Asr assignment 2023. `https://github.com/ZhaoZeyu1995/asr_assignment/`, 2023. Accessed: 2 July 2023.

Appendix A

Detailed Experimental Results

A.1 Speech in a Quiet Environment

Table A.1: Aggregated WER of speech samples in a quiet environment

Codec	Sampling Rate (kHz)	Bit Rate (kbps)	Word Error Rate	
			Google Cloud	Rev AI
(Uncompressed)[1]	48	768	.0391	.0329
(Uncompressed)[2]	8 (Resampled)	128	.0434	.0419
GSM	8	13	.0568	.0514
G.729	8	13	.0642	.0530
G.726	8	16	.0617	.0519
		24	.0508	.0459
		32	.0447	.0435
AMR	8	4.75	.0792	.0614
		5.15	.0767	.0599
		5.9	.0692	.0541
		6.7	.0660	.0534
		7.4	.0624	.0530
		7.95	.0613	.0507
		10.2	.0541	.0460
		12.2	.0522	.0476
Speex	8	2.15	.2879	.1991

[1] Original audio was retrieved in 16-bit LPCM and stored in `.wav` containers

[2] Audio was resampled and encoded as 16-bit LPCM stored in `.wav` containers

Speex	8	3.95	.1371	.0964
		5.95	.0963	.0686
		8	.0753	.0567
		11	.0594	.0491
		15	.0511	.0461
		18.2	.0475	.0443
		24.6	.0452	.0440
Opus (`audio` application)	8	6	.0997	.0569
		8	.0607	.0483
		10	.0518	.0453
		12	.0503	.0433
		16	.0463	.0426
		24	.0443	.0418
		32	.0426	.0415
Opus (`voip` application)	8	6	.0754	.0523
		8	.0581	.0477
		10	.0515	.0452
		12	.0489	.0435
		16	.0463	.0428
		24	.0456	.0421
		32	.0449	.0420
MP3	8	8	.1245	.1687
		16	.0599	.0464
		24	.0513	.0417
		32	.0500	.0407
AAC	8	8	.1404	.1997
		12	.0708	.0796
		16	.0567	.0486
		20	.0507	.0437
		24	.0478	.0444
Codec 2	8	.45	.4869	.4645
		.7	.4433	.3679
		1.2	.1975	.1319
		1.6	.2146	.1292

Codec 2	8	2.4	.1237	.0853
		3.2	.0931	.0656

A.2 Speech in Simulated Noisy Environments

Table A.2: Aggregated WER of speech samples in simulated noisy environments

Codec	**Sampling Rate (kHz)**	**Bit Rate (kbps)**	**Word Error Rate**	
			Google Cloud	**Rev AI**
(Uncompressed)[3]	48	768	.1367	.0981
(Uncompressed)[4]	8 (Resampled)	128	.2086	.1327
GSM	8	13	.2420	.1577
G.729	8	13	.2281	.1821
G.726	8	16	.2905	.1785
		24	.2451	.1477
		32	.2207	.1377
AMR	8	4.75	.2808	.2209
		5.15	.2692	.2133
		5.9	.2455	.1938
		6.7	.2347	.1855
		7.4	.2268	.1741
		7.95	.2284	.1738
		10.2	.2105	.1599
		12.2	.2079	.1531
Speex	8	2.15	.5730	.4539
		3.95	.3842	.2968
		5.95	.3126	.2350
		8	.2647	.2043
		11	.2311	.1667
		15	.2104	.1484
		18.2	.2155	.1401
		24.6	.2068	.1349

[3] Original audio was retrieved in 16-bit LPCM and stored in `.wav` containers

[4] Audio was resampled and encoded as 16-bit LPCM stored in `.wav` containers

Opus (`audio` application)	8	6	.3513	.2883
		8	.2446	.1954
		10	.2027	.1716
		12	.1894	.1550
		16	.1724	.1437
		24	.1893	.1362
		32	.1865	.1329
Opus (`voip` application)	8	6	.3202	.2217
		8	.2358	.1778
		10	.1992	.1670
		12	.1869	.1506
		16	.1714	.1497
		24	.1656	.1401
		32	.1655	.1374
MP3	8	8	.3808	.3405
		16	.2363	.1533
		24	.2053	.1388
		32	.1994	.1330
AAC	8	8	.3237	.2795
		12	.2270	.1763
		16	.2578	.1495
		20	.2444	.1424
		24	.2342	.1403
Codec 2	8	.45	.7884	.7762
		.7	.7606	.7113
		1.2	.5436	.4751
		1.6	.5541	.4845
		2.4	.4109	.3584
		3.2	.3607	.3110