Bachelor Degree Project

An Analysis of Data Compression Algorithms in the Context of Ultrasonic Bat Bioacoustics

Authors: Max Anderson,
Benjamin Luke Anderson
Supervisor: Ilir Jusufi
Semester: HT 2022
Subject: Computer Science

Abstract

Audio data compression seeks to reduce the size of sound files, making them easier to store and transfer, and is thus a highly valued tool for those working with large sets of audio data. For example, some biologists work with audio recordings of bats, which are well known for their frequent use of ultrasonic echolocation, and so these biologists can accrue massive amounts of high frequency audio data. However, as many methods of audio compression are designed to specialize in the more common range of frequencies, they are not able to sufficiently compress bat audio, and many bat biologists instead work without compressing their data at all. This paper investigates the desiderata of a data compression method in the context of bat biology, experimentally compares several modern data compression algorithms, and discusses their pros and cons in terms of their potential use across various relevant contexts. The paper concludes by suggesting the algorithm Monkey's Audio for machines able to handle the higher resource demands it has. Otherwise, FLAC and WavPack yield similar size reduction rates at a significantly faster speed while being less resource intensive. Of note is the interesting result produced by the algorithm 7-ZIP PPMd Solid, which achieved consistently outstanding results within a single dataset, but its generalizability has yet to be determined.

Keywords: Bioacoustics, data compression, bats, audio data, ultrasonic

Contents

1. Introduction

This paper is a 15 HEC Bachelor thesis in Computer Science, primarily investigating the area of audio data compression. Within this area, special attention is paid to the context of how data compression operates in the context of biology, and more specifically how compressing the audio recordings of bats impacts the compression algorithms and their performance. Bat audio introduces unique data concerns, as their vocalizations are orders of magnitude higher in frequency than what most compression methods are intended to accommodate [1].

As a means of catering to the common use cases of storing the sounds of either music or human speech, many of the current digital audio compression methods overwrite or discard any data that falls outside of the human-audible range [2]. However, these heuristics can be detrimental when compressing audio outside the limits of normal human hearing. This can lead to critical data in fields like biology being mistakenly discarded as useless noise [3]. Moreover, the high frequencies being recorded during the study of bat echolocation necessitates high quality recordings with high sampling rates, leading to the generation of unusually large amounts of data during the study of bat cries and therefore greater concern for the efficiency of data storage and transfer [4].

This paper intends to elucidate the desiderata for audio file compression for this context and experimentally determine which method or methods are most appropriate, focusing on the demands largely unique to high frequency sounds common in chiropterology, the study of bats.

1.1. Background

Within the field of bioacoustics, defined as the study of sounds made by animals, it is considered standard to work with audio recordings of animals using computers to store, transfer, and analyze the data [4]. In order to study the behavior of animals over time, bioacousticians frequently record for several hours per night for several nights or even weeks, often using multiple recording devices in different locations [5]. This generates multiple terabytes of audio which is then stored, duplicated, and transferred in its entirety multiple times. This results in large amounts of time, money, and energy being spent on computer processing power and storage space. This is exaggeratedly problematic when recording animals whose vocalizations are of such high frequency [3].

This is due to how most audio compression methods determine what data within the file should be considered air, which in this context means data not needed to represent the file's critical information. Frequently found within the structure of modern audio compression algorithms are components responsible for identifying the data within the file that corresponds to sound that cannot normally be heard by humans, meaning sounds with frequencies above 20kHz [2]. In most cases, manipulating this data does not produce any meaningful changes to the end result, as audio files are only expected to be heard by people with no other use [2].

However, many species of bats, dolphins, and insects have ultrasonic vocalizations, which means that these algorithms would incorrectly identify those sounds as air. Additionally, in order to properly record high frequency audio with sufficient precision,

a high sampling rate is needed, meaning that the amount of data in the average ultrasonic audio file is significantly more than standard audio files of the same length [4]. Higher quality recordings are also frequently used in order to make it possible to properly separate sounds from surrounding environmental "noise" [6]. Thus, many experts within the field of bioacoustics must either find expensive or impractical means of working with massive amounts of data or accept that the resulting audio will contain only a small fraction of the original data.

As recording hardware and audio analysis software grow more powerful and convenient, the field of bioacoustics similarly grows in popularity [7]. However, without a corresponding improvement in data handling, file transfer time and storage space may prove to be ever more limiting factors, potentially leading to prohibitively expensive hardware being deemed necessary for working within the field.

1.2. Related Works

As recently as 2021, Heath et al. [3] reviewed the effects that modern lossy audio data compression methods have on data accuracy within the field of ecoacoustics, which is a closely neighboring field to bioacoustics. The study concluded that, should data size become a larger concern than perfect data accuracy, the most efficient means of lossy data compression that yields tolerable accuracy loss can reduce the size of a file to approximately a quarter of the original size. However, it notes that "lossless compression is always desirable", and so it presents these methods as tradeoffs for those unable to handle the amount of data even after lossless compression, thus deferring the evaluation of these lossless algorithms, which this paper aims to address. Additionally, it provides a methodological framework from which audio compression algorithms may be experimentally measured and compared.

Beurden [8] also performed an experimental comparison between several audio compression algorithms in 2015. Examining exclusively lossless algorithms, this study measured the metrics of compression speed and proportional file size reduction when given audio files of varying lengths and sample rates. The set of tested audio is entirely composed of music, meaning that while the study offers further precedence for performing such an experiment, its results cannot be generalized to the context of this project.

Further work done by Hidayat et al. [9] and Chen and Yu [10] also contribute to the standard for experimentally measuring and comparing the metrics of audio compression algorithms. The former [9] details in great depth both the implementation of its methodology for performing the experiment and the mechanisms of the algorithms it examines, those being Huffman and Arithmetic. The latter [10] performs a similar experimental analysis on genetic algorithms, thus providing a foundation for evaluating novel audio compression methods in comparison to existing ones. These studies exclusively utilize music files for their test data sets, inhibiting the results from being confidently applicable to ultrasonic bioacoustics and further exemplifying the bias for human audibility in audio compression methods.

1.3. Problem Formulation

As previously mentioned, human audibility bias is pervasive throughout the field of audio data compression [2], [8], [9], [10]. Consequently, little consideration has been given to the frequencies that lie outside the comparatively narrow range of human perception. Ultrasonic sound, defined as the range of frequencies that are too high for humans to hear, has been neglected when it comes to data compression despite being fundamental in numerous fields of study and application. This has resulted in many compression methods fundamentally relying on practices that cannot accommodate characteristics unique to the ultrasonic range [2]. Therefore, the aim of this paper is to answer the following research question:

What are the best performing audio data compression methods in the context of ultrasonic bat bioacoustics?

To fully define, contextualize, and answer this question, several sub-questions will be explored throughout the course of this paper:

SQ1. What are the desiderata for an audio compression method from the perspective of biologists specializing in ultrasonic frequencies?

As this project is focused on the use case of bioacoustics, the exact nature of the audio can best be described and understood by those with the most experience in that field. Similarly, the specifications for what a compression method should achieve to best suit this audio should be catered to the requirements of those within the field.

SQ2. How well are these desiderata satisfied by currently available methods?

When analyzing the methods of audio compression that are readily available, how well do they fulfill the requirements elicited in the previous question? Which metrics fall short of what is considered necessary and to what extent are they insufficient, and which requirements are either fully fulfilled or completely ignored?

SQ3. Which of the examined methods can be recommended for use by the target group within the various contexts they operate in?

Finally, once the previous questions have been sufficiently explored, the methods and their performance should be contextualized by their potential use. In other words, how can the results of SQ2 be interpreted and applied by a given member of the target group, considering the varying use cases they may encounter?

1.4. Motivation

While the bias favoring human perception in data compression makes sufficient sense in most contexts, the limited consideration given to exceptional cases has created substantial problems in important areas of scientific research. Those working in the field

of bioacoustics, as well as many other fields which commonly interact with ultrasonic frequencies, consequently consider many of the most popular and supported means of data compression unfeasible. This results in them having to spend a large number of resources dealing with massive, uncompressed files or losing significant amounts of critical data in the process of compressing them. Making this problem worse is the high sampling rate required to record the high frequencies that define ultrasonic sound, substantially increasing the amount of data generated. The storage space this data requires can easily reach several terabytes within a few weeks, even for smaller projects.

Eliciting and understanding the best way to compress audio files that do not explicitly prioritize human hearing would make analyzing ultrasonic sounds easier, faster, and cheaper, resulting in more efficient progress to be made in those fields, and would lower the barrier of entry to those fields drastically [11]. Additionally, outlining the theoretical deconstruction of modern compression methods and detailing its refinement for a specific scientific context can provide a solid foundation for others to do similar work for other contexts.

1.5. Results

As the goal for this project is to experimentally determine the best audio data compression algorithm within the context of ultrasonic bioacoustics, the specifics of what makes one algorithm better than another must be defined. Following the precedent set by previous works [3], [8], [9], [10], three metrics have been identified as critical for comparing any examined method:

1. **Accuracy**: The percentage of data that is not lost or corrupted during compression
2. **Size Reduction**: The ratio between a file's size after compression and its original uncompressed size
3. **Speed**: The amount of time taken to compress a given amount of data

Accuracy is measured by comparing the value of all bytes within a file before compression to the bytes present after decompression. An accuracy of 100% indicates that every byte from the original file has an identical value to its decompressed counterpart, with no additional bytes present. Size reduction, also referred to as compression ratio, is defined as the percentage size of the compressed file in relation to the size of the original file [12]. For example, if the original file has 400 bytes and the compressed file has 100 bytes, the compression ratio would be 25.0%. Speed is simply the number of seconds it takes for a given algorithm to fully compress a file.

A set of selected algorithms will be tested by compressing a set of audio files representative of what is used by the target group and measuring the metrics listed above. These metrics will be analyzed and prioritized in order to establish a quantifiable evaluation with a sufficient comparison.

1.6. Scope

This paper and the work discussed within focuses almost exclusively on audio recordings of bats. While there are many animals with ultrasonic vocalizations, the extremely broad range of frequencies that is covered by all these animals as well as the variations in how this data is gathered and analyzed produces too wide of a scope to be addressed by this project. While it may be possible that the results produced are applicable to recordings of other animals or other audio common in other fields, the work done in this project pays no consideration to how these results perform in those contexts.

1.7. Target Group

This work primarily focuses on the context of recording and analyzing audio recordings of bats, and therefore biologists and bioacousticians that specialize in chiropterology are the main target group. Generally, the specific mechanics of data compression are considered to be too complicated and too niche to be commonly understood by those outside the field of computer science. As such, it is assumed that the target group has at most a casual knowledge of data compression.

1.8. Outline

The remainder of this paper details the study and work performed by this project in the pursuit of answering the previously defined research questions. Section 2, *Method* describes the methodology of the paper, which includes a brief overview of how data will be collected and the framework used to ensure that the project is done ethically and its results are valid. Section 3, *Theoretical Background* presents the aggregation of the performed literature review, isolating the information deemed critical to understanding the proceeding sections of the paper. Section 4, *Research project – Implementation* provides an in depth explanation of the steps taken to gather the data necessary for answering the research questions. Section 5, *Results* lists the data gathered by following the steps defined in Section 4 while avoiding drawing conclusions until further sections. Section 6, *Analysis* highlights statistical patterns found within the data and contextualizes them. Section 7, *Discussion* describes the deeper mechanisms or implications of the results that were found to warrant such an explanation. Section 8, *Conclusions and Future Work* fully connects the results to the research questions and uses these connections to propose answers. Additionally, Section 8 mentions several means of potentially continuing the work done in this project and how this paper would support such work. Finally, Sections 9 and 10, *References* and *Appendix* respectively, list the outside sources of information that this paper relies on as well as any documentation created by this project that was deemed unnecessary to report on in its entirety.

2. Methodology

The purpose of this section is to describe the methods by which the research questions in the previous section will be analyzed and answered. An overview of the techniques and methods that will be used are given, with each method being discussed in greater detail thereafter. Finally, the issues of reliability, validity, and ethics are discussed.

2.1. Research Project

Beginning with the goal of answering SQ1, the first topic to consider is what sort of algorithm is desired. More specifically, what features are required, what qualities are desired, and what behavior needs to be avoided.

To this end, a small number of interviews will be conducted with experts of various kinds who regularly work with ultrasonic bat audio. Care will be taken to make contact both with academic researchers who perform in-depth analysis on ultrasonic bat audio as well as industry professionals whose primary concern is broad, large-scale data collection for the purpose of performing censuses.

These experts will be asked a series of questions about their use of ultrasonic bat audio data, with the intention of understanding the type and amount of data they frequently work with, how it is used, what they consider to be the most valuable information in the data, and what they would value in a compression algorithm. The questions as well as the interviews are available in Appendix A.

Once the desiderata for a compression algorithm have been established from these interviews, a number of audio compression algorithms will be tested to see how well they satisfy the requirements, thus answering SQ2. To this end, each algorithm will be measured on how well it can compress sample data collected from the interviewees by request. A literature review will also be conducted during this stage to better understand the performance of each algorithm and gauge whether the experiment's data would be generalizable. Finally, to answer SQ3, the collected data and knowledge will be culminated into brief, qualitative descriptions of the methods that are found to be recommendable in respect to one or more contexts.

2.2. Research Methods

2.2.1. Interviews

With the objective of eliciting the desiderata referenced in the research questions, members of the target group will be contacted with a request for an interview. Following the conceptual framework and design principles outlined by Flick [13], these interviews will ask a suite of questions designed to produce an overview of the field's current practices regarding data handling as well as the requirements that will define the desiderata moving forward. For health and logistical reasons, interviews will be conducted via online video calls, and additional questions may be asked in cases where a response to a prepared question prompts further inquiry.

In cases where a participant expresses interest in answering questions, but cannot or will not be available for an online video call, the questions will be sent as text via email,

and the responses will be gathered via email as well. Any follow up questions, additional clarification, or further communication shall likewise take the form of text-based emails.

2.2.2. Literature Review

Once the performed interviews have adequately provided an understanding of the context that the research questions exist within, a deeper knowledge of the processes and concepts that fundamentally comprise the problem and its potential solutions shall be developed. To this end, relevant academic and peer-reviewed scientific literature will be aggregated and studied, with all sources of prominent background information being explored in the *Theoretical Background* section of this paper. This paper shall refrain from the comprehensive extent to which a paper exclusively performing a literature review would present its findings in order to communicate only the knowledge required to understand and interpret the contribution of this project. With this said, a proper understanding of the subject is critical for making and understanding meaningful contributions to it [14], and so this review seeks to ensure the novelty and accuracy of this project's work.

2.2.3. Controlled Experiment

As outlined previously in Section 1.5, *Results*, the metrics deemed most critical for comparing various data compression algorithms are accuracy, size reduction, and speed. In order to accurately and reliably measure these attributes for multiple compression algorithms across a variety of contexts, controlled experimentation shall be conducted. Drawing from the foundational contribution of previous works [3], [8], [9], [10], each algorithm shall be tasked with compressing an exact copy of a set of data files which are collected by request from members of the target group so that they are sufficiently representative of this paper's scope . After several iterations of this process for each algorithm using multiple datasets, ensuring that all external variables are kept static, the resulting measurements shall be statistically analyzed. Finally, the aggregated results and the knowledge derived from the literature review shall be used to answer the latter two research sub-questions.

2.2.4. Considered Methods

Initially, surveys were considered as a means of forming a general perspective of the context that the target group operates within, as well as to appraise the impact of the problem. However, this was eventually dismissed in favor of the previously discussed interviews, as the target group lacks a medium of mass contact, e.g., an online forum with significantly many active members. This means that surveys would have to be presented to participants individually, at which point the surveys would no longer possess its main advantage over interviews, that being more efficient administration [15].

2.3. Reliability and Validity

As with any scientific procedure, the standardization of testing methods and data collection contributes the bulk of this paper's reliability [16]. Therefore, the steps,

equipment, and testing environments used within this project shall be detailed thoroughly and precisely to ensure that the results presented in this paper are reproducible.

Certain metrics common within computer science can vary greatly based on factors that often go undetected. For example, the speed of compression can be slowed by any background operations the machine quietly performs during only one test iteration [17]. As such, the validity of the gathered test data relies on controlling the test environment beyond simply using consistent hardware and software. Therefore, the design and implementation of the experiment shall be constructed with special attention to the heightened need for a controlled environment. Additionally, an increase in iterations per test suite and a critical eye for outliers in the resultant data both serve to diminish the impact of this issue.

2.4. Ethical Considerations

As will be elaborated on in future sections, much of the work performed by the target group falls under the category of either scientific research or ecological consulting. As such, the conclusions of their work are regularly validated, discussed, and challenged by others in their respective fields. Should the data that these conclusions are based on be corrupted by some error present with the compression algorithm, the conclusions may be unverifiable, and the work would consequently be wasted. Therefore, any algorithm explored in this paper should be held to the same reliability standards as those set by the scientific community to avoid inadvertently sabotaging the works of others.

Furthermore, to ensure that participants of interviews are not misled and that the inclusion of their input in this paper is consensual, any and all requests for interviews shall begin by explaining the context and goals of this paper and their potential inclusion in it. Additionally, care shall be taken to confirm the participants' willingness to have their answers and data shared publicly in this context both before and after their interview [13]. Moreover, interviewees have been offered full confidentiality to protect their identities.

Finally, to avoid plagiarism and theft, all works both theoretical and practical that this project uses as a foundation, inspiration, or support shall be handled with care and respect. This includes obtaining all software legally, abiding by their license of use, and properly crediting their source whenever applicable.

3. Theoretical Background

3.1. Bats

Bats, classified as mammals within the order Chiroptera, commonly possess the ability to echolocate, meaning that they measure the distance between themselves and other organisms or objects by using sound [18], [19], [20]. They do this by emitting ultrasonic vocalizations at frequencies up to 212 kHz [21] and gauging the delay until the resulting echoes are heard, as a longer delay means the sound had to travel a correspondingly longer distance. The primary reason that most species of bats echolocate using ultrasonic frequencies is that their diet consists largely of small insects, and so if the wavelength of the sound is too great, the sound would not provide enough precision for the bats to efficiently hunt [18].

Bats are considered to be important biodiversity indicators due to their global distribution and their relatively high population sensitivity, which means that disruptions to an ecosystem have a relatively strong effect on local bat populations [20], [21], [22]. As a result, measuring bat population growth and decline provides valuable insight into the general health of an ecosystem, including the impact of climate change, habitat loss, and disease on the area [21], [22].

3.1.1. Bat Census

Census groups collect and analyze data with the intention of identifying the current condition of wildlife within a given location [22]. This can include validating the presence or absence of certain animals, estimating the population of various species, or identifying potential risks of planned environmental or construction projects. Commonly encountered within this work is the need to survey a wide geographical area with minimal interaction with that area to ensure data validity.

Once bat audio has been recorded, it is generally analyzed by means of a number of automatic detection programs. Such tools are frequently used to process the file–for instance to apply a high-pass filter or reduce noise [6]–to detect bat sounds within an audio file, and to identify the number and species of bats in the recording [6], [19], [20], [22], [23]. However, call structures can vary significantly within a species, making species identification difficult [21]. As Rydell et al. [23] discuss, manual validation of the results of these programs is necessary, as the reliability of species identification can vary wildly.

Species identification is often most concerned with information in the recording such as the frequency of the highest amplitude sound, the minimum and maximum frequency, and the timing and duration of the call [18], as well as its harmonics [21].

3.1.2. Bat Research

Bat researchers, otherwise known as bat biologists, regularly monitor the condition and behavior of bats, both in the wild and within highly controlled experimental test cases. In both contexts, the analysis of the bats' vocalizations provides valuable insight into their behavior and health, and can also act as the primary focus for academic study. Depending on the type of research, bat audio recordings can contain a wide variance of background noise [19]. If studying bats in the wild, sounds of the wind, weather, and

other animals can be pervasive [19], while bats being studied in a laboratory setting can have little to no background noise in their recordings at all.

While many of the same analysis programs mentioned in Section 3.1.1 are also employed during Bat Research, species identification is not always needed, especially in the case of laboratory research where the bats being studied are known. While the exact metrics vary significantly based on the research being done, much of the same information used in census work is still used for research.

3.2. Audio Recording

3.2.1. Sampling and Digitizing

In order to represent real-life analog audio signals as digital signals that can be handled by computers, they are converted into bounded, discrete-value sequences [24]. This process involves two steps, sampling and digitizing, which both act to quantize the data or to approximate the data by restricting its values to a discrete set of values [24]. Sampling an audio signal means taking a measurement of its value at discrete points in time, and digitizing means converting that measurement to a digital value [24]. A digital signal is therefore the set of these digitized values at each of the sampling points. Thus, as the number of sampling points increases, the amount of data needed to represent the original audio signal increases [24], [25].

Additionally, the samples of recordings can be stored at differing levels of granularity, often expressed in terms of the number of bits per sample [25] or "bit depth" [26]. For high-quality audio files, this is frequently 16 bits per sample, though other values are possible. Higher bit depth also increases the size of the resulting file [25], [26].

3.2.2. Nyquist-Shannon Sampling Theorem

According to the Nyquist-Shannon sampling theorem, in order to reconstruct an analog audio signal from its digital signal, the frequency at which the signal was sampled must be greater than double the frequency of the sound being represented [24], [26], [27]. In the case of complex signals that must be represented by means of a Fourier transform (discussed below), the sampling rate must be higher than double the frequency of the highest non-zero signal in the transform [27]. See Figures 4.1 - 4.4 below for examples of the effects of different sampling rates above and below this threshold.

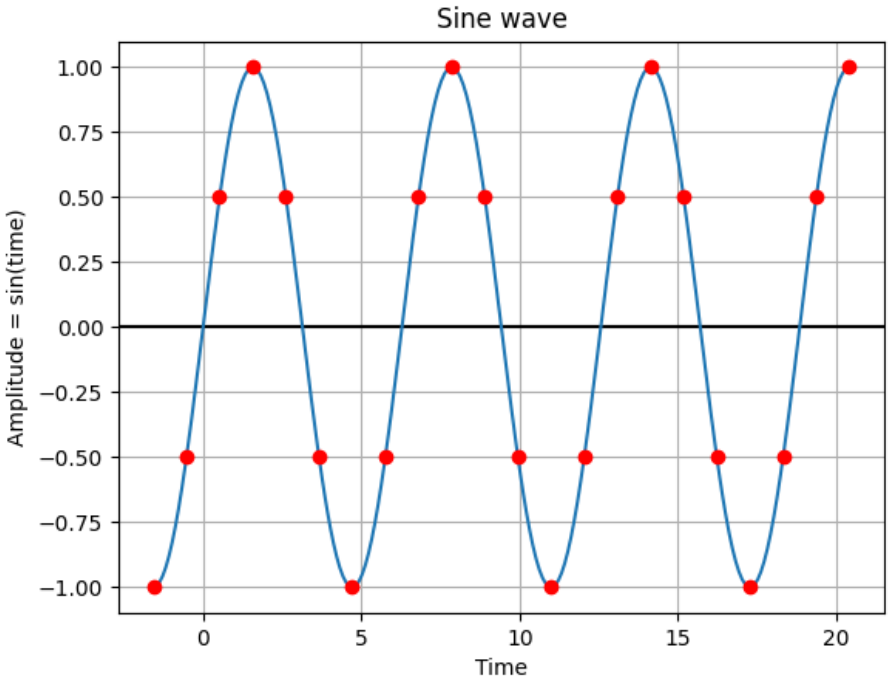


Figure 4.1: The samples (red dots) form a digital signal, which is enough to deduce the original analog wave's properties.

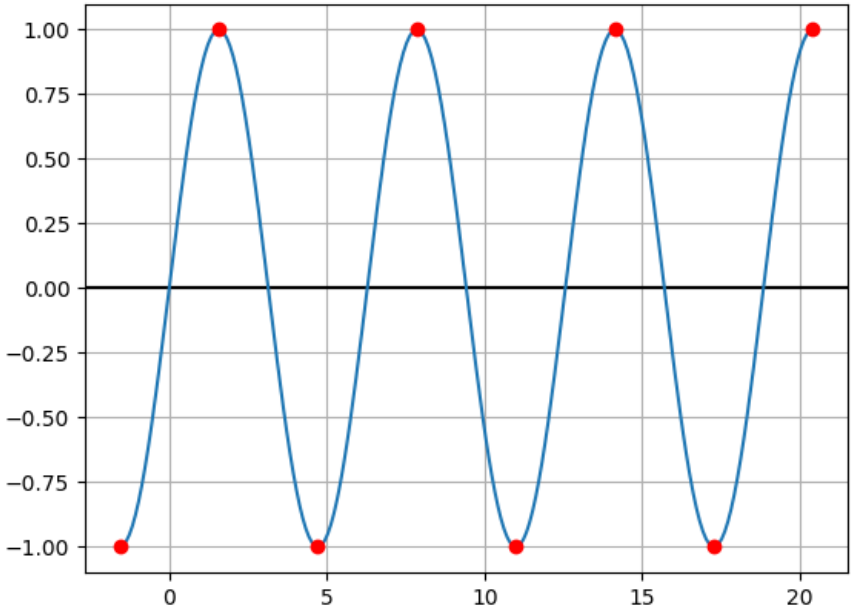


Figure 4.2: A sampling rate twice the original signal's frequency is the minimum rate which is still capable of capturing the properties of the original analog signal.

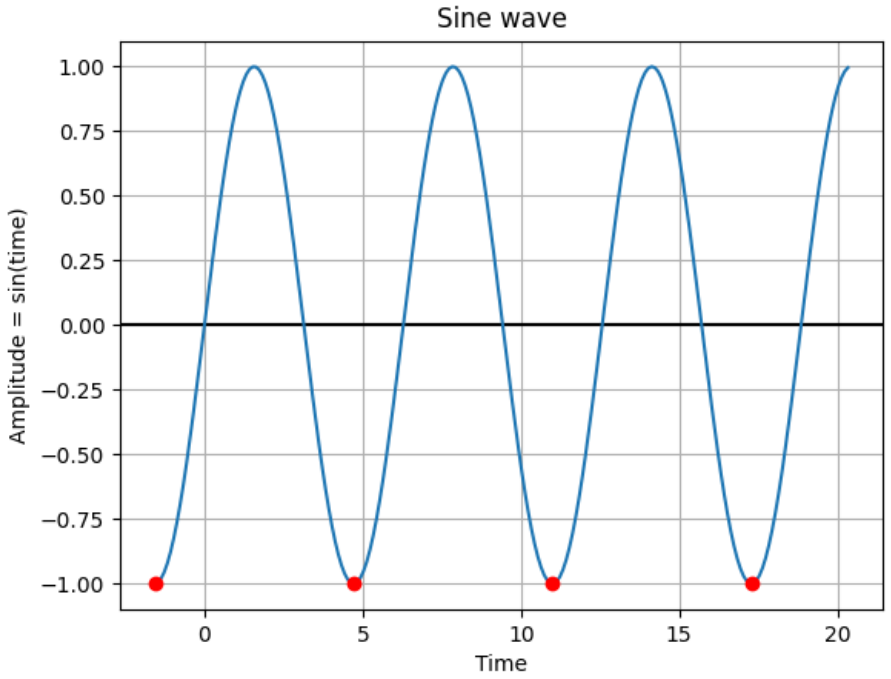


Figure 4.3: At a sampling rate equal to the signal's frequency, the signal is indistinguishable from a flat line.

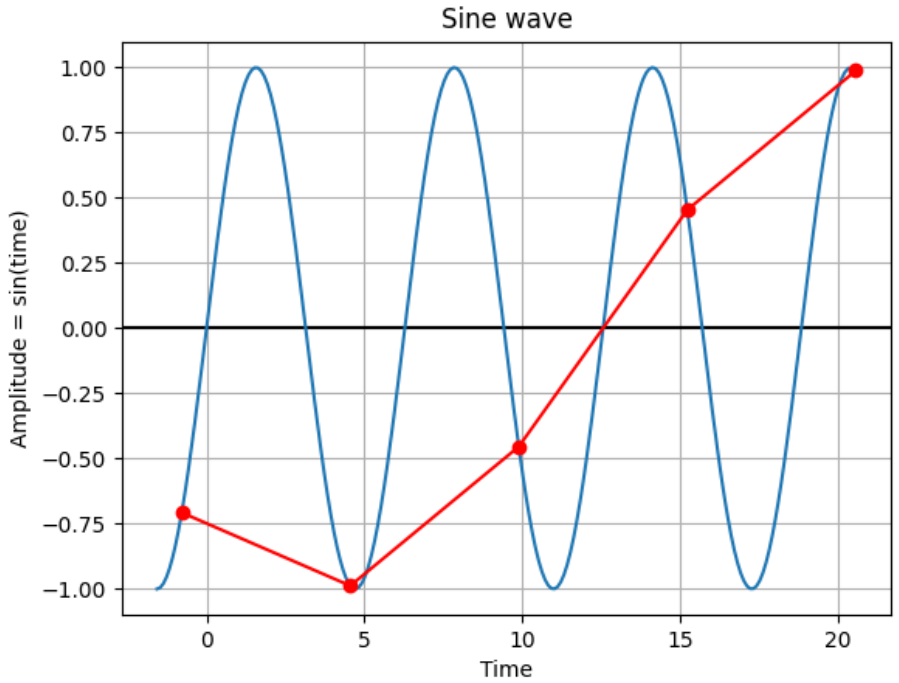


Figure 4.4:At a sampling rate less than the signal's frequency, the digital signal can start resembling entirely different frequencies than the original signal.

3.3. Data Compression

Information theory dictates that it is impossible to create a compression algorithm capable of reducing the size of any file without losing data in the process [26]. This can be expressed as a consequence of the pigeonhole principle, which is the mathematical principle which states that "if *n* objects are distributed over *m* places, and if $n > m$, then some place receives at least two objects" [28]. For the case of data compression, this is realized by the fact that there exist 2^n different uncompressed files of the size *n* bits, but only 2^n-1 possible smaller files, meaning it is impossible to map uncompressed files to smaller compressed files such that each uncompressed file will generate a unique compressed file. This inability to achieve an injective mapping to a smaller set means that it is impossible to both compress all files to a smaller size while simultaneously being able to recover the original, uncompressed files from their compressed versions. One must either settle for a compression algorithm which is unable to be fully reversed (thus it loses data, which is called lossy [29]) or one must accept the fact that one's lossless algorithm will be unable to reduce the size of all files, and indeed may even increase the size of certain input files [26].

Phrased in a more intuitive way, if one could use the same algorithm to losslessly compress any file, then it would be possible to continuously and recursively compress any file down to a size of nearly zero, which is clearly an impossibility.

As a result of the above law of information theory, compression algorithms are tailored to their intended purpose [26], [27]. Lossless algorithms are tailored so that they decrease the size of files considered important in the hopes that the files which grow in size are unimportant. Lossy algorithms, which have an easier time achieving file size reduction due to the fact that they throw away unimportant data, must still be tailored in order to define what data is important.

3.3.1. Model and Code

The tailoring of an algorithm to a specific purpose is expressed through its two conceptual components: its *model* and its *code*. The model of a compression algorithm is an extraction and expression of redundancies that exist in the data to be compressed [26], [27]. These redundancies are often domain- and medium-specific; audio of human speech will feature radically different patterns than those found in the image of a bird.

An algorithm's *code*, on the other hand, is the assignment by which symbols or sequences in the original data are stored in compressed format [26], [27]. For instance, one could encode a series of coin flips by using 0 for heads and 1 for tails, or one could encode these as fully spelled out strings *Heads* and *Tails*. Generally speaking, the more common a symbol or sequence is in the source file, the fewer bits a compression algorithm should ideally use to represent it [26], [27].

3.3.2. Solid vs Nonsolid Archives

One major choice in most compression techniques is whether the compressed archive should be solid or not. In solid archives input data is compressed together, and patterns which are found in one file can be reused or refined to efficiently compress other files as well. This generally achieves more efficient compression ratios because patterns identified in one file can be reused and refined to efficiently compress other files. One

major downside to this is that it can become difficult to extract a single desired file from the block, at worst requiring that the entire block be decompressed before the file can be restored [27].

However, in nonsolid archives, the input data is separated into evenly sized chunks or blocks which are compressed separately. This can be a faster process because generally fewer patterns are established within less data, patterns will only exist at smaller distances from each other (requiring less searching for matching patterns), and because compressing multiple blocks can occur simultaneously by using multiple processing threads. It's also slightly more robust, since if data becomes corrupted then the damage may be contained to that block, whereas in a solid archive the entire thing may be lost [27].

3.3.3. Audio Codecs

Audio codecs are encoding/decoding techniques which express audio signals as a code [2]. Modern audio codecs almost invariably feature data compression techniques as a part of their operation, with the terms codec and compression algorithm being used synonymously much of the time [2], [25]. These compression techniques often assume that the input is human speech or music [2], [27]. As a result, audio compression methods are often centered around a psychoacoustic model, that is to say a model of the way sound is received and interpreted by the human ear [2], [27], [30]. This so-called perceptual coding [2], [25], [26] or perceptive coding [30] identifies irrelevant information in the audio and incorporates specialized techniques such as critical band analysis [2], M/S stereo coding [26], joint frequency encoding [27], spectral masking [27], and temporal masking [2], [25], [26], [27] to take advantage of the unique properties of audio files.

For instance, in the case of multi-channel or stereo audio, separate records are kept of what should be heard by the left ear versus the right ear. However, these channels rarely differ by a large degree, and so an audio codec may attempt to compress this information by combining the two records into one and separately saving stereo data about the ways in which the channels differed [2], [26], [27].

3.3.4. General Compression Algorithms

General compression algorithms are algorithms that make few assumptions about the nature of the data being compressed [26], [27]. Many transform the data first into something simpler or easier to encode.

The two types of transforms used in this study are dictionary based and context based. Dictionary-based transforms search for commonly repeated sequences and replace these with shorter references to the sequence, storing a dictionary of references and their meanings alongside the encoded data [26], [27]. Context-based algorithms attempt to adaptively build an understanding of the file as it is parsed in order to build more precise probability models and thus predict future symbols [26], [27].

4. Research Project – Implementation

4.1. Interviews

Interviews were conducted with researchers and professionals working with bat ultrasonic audio data over the period from March 8th to March 17th. Contact was initiated via an email providing context that described this project and how their participation would be incorporated.

In order to properly contextualize the results gathered during the interviews, the interviewees will be categorized based on their normally expected interactions with and uses of bat audio. These categories, which are expanded upon in the following subsections, are intended only to summarize the contexts and patterns that were found to be common during the performed interviews, and are not meant to be representative of any group, field, or occupation outside of the scope of this paper. For the purposes of this paper, the interviewees in Interview A and C are categorized as *Bat Census*, and the interviewee in Interviewee B are categorized as *Bat Research*.

4.1.1. Questions

A list of twelve questions was created to be used in the interviews. These questions were designed to yield insight into the context in which the interviewees encountered the problem of big data. Additionally, the questions sought to elucidate the impact of the problem, how the interviewees addressed it, and what metrics would be considered necessary or desired in a potential solution. The questions and their rationale are as follows:

1. Briefly, what specifically is your job title/description, how long have you been doing it, and what is your current project/goal?

This first question acts as an introduction to provide context for the participant's answers and how representative of their field they may be.

2. On average, how much bat/ultrasonic audio data do you generate for a project? How large are individual files on average/at most?

Question 2 serves to quantify the foundational concern of this paper, that being the amount of data that is generated in the target scope.

3. How impactful is the size/amount of audio data that you work with? Does storage or transfer of these files consume a problematic amount of storage space or time?

Similarly to the previous question, this question aims to qualify the impact that this amount of data has on those in the target group.

4. What software do you use to analyze this audio, and what functions do you primarily use it for? What can/can't be done automatically?

Question four directly connects to SQ1 by determining what is needed both by and from the audio in a computer science context.

5. What percentage/how much of the average audio file is empty/useless/"noise"? How do you identify this noise? Is this noise removed somehow before/during analysis?

The idea of noise pertains to the metric of accuracy in the sense that the loss of data that represents unneeded sound could be potentially manipulatable with little concern.

6. Does ultrasonic audio data have to be treated significantly differently from other audio data?

Once again directly relating to SQ1, this question intends to discover any feature or behavior of ultrasonic audio data that would be unintuitive to someone outside of the target group's field.

7. In what format do you generally store bat/ultrasonic audio data (e.g., full spectrum vs zero crossing, compressed vs uncompressed) and what file types do you commonly use (e.g., wav, zc, mp3, flac, zip)? Why?

Relating to SQ2, this question attempts to gather potential data compression algorithms for experimentation while ensuring their contextual relevance.

8. How long do you generally store any given bat audio data and how often do you generally transfer significant amounts of it?

This question assists in prioritizing the metric of size by quantifying the degree to which a smaller file size would be beneficial.

9. Would you value a way to reduce the size of the bat/ultrasonic audio data you work with?

Serving something of a dual purpose, this aims to gather further understanding of the target group's perception of the problem and to clarify the intention that these questions are asked with.

10. Three important metrics for determining the value of a compression algorithm are:
    - Size: How much smaller is the compressed file compared to the original?
    - Speed: How much time does it take to compress a file?
    - Accuracy: How much detail was lost during compression?

    How do you value these metrics? Are there other metrics that are also important to you?

Question 10 asks for the target group's prioritization of the metrics that are used to compare the compression algorithms.

11. Do you know whether anybody has developed a data compression method focused on ultrasonic audio before? If not, do you know why not?

Depending on the answer provided to this question, it either assists in finding more algorithms for experimentation or it supports the novelty of this project.

12. It has been rumoured that zero crossing files may be phased out in the future. What have you heard about this, if anything?

Zero crossing files were at one point considered to be included in the experimental portion of this project, but were eventually considered outside of the scope. This was due to the significant difference between use cases in which zero crossing files are common and one where .wav files are common. Nevertheless, question 12 intends to address SQ2 by asking if zero crossing files are worth considering even in their own context.

4.2. Experiment

In service of measuring, analyzing, and comparing the performance of different audio compression methods to address SQ2, *How well are these desiderata satisfied by currently available methods?*, four sets of uncompressed audio files in the .wav format were constructed from data provided by the target group. These files were screened to confirm that the data was representative, uncorrupted, and not duplications of other files in the sets. Upon execution of the test, various audio compression methods were instructed to compress each set, creating compressed copies of the data without altering the original set's files. Each method was invoked sequentially, meaning that no two methods were ever running concurrently, so that the results of each performance were sufficiently isolated.

The audio data selected to be compressed via the execution of the experiment was divided into four sets. These sets, hereby referred to as **Dataset A**, **Dataset B-1**, **Dataset B-2**, and **Dataset B-3** respectively, consist of uncompressed .wav files where each file is present in exactly one set. Files were organized into sets based on the date of their original recording, so that all files in a given set were recorded no more than three days apart from each other, and were recorded using the same machine.

Additionally, **Dataset A** is distinct from the other datasets in terms of the content of its recording. **Datasets B-1**, **B-2**, and **B-3** are meant to represent the use case common to bat biologists, and therefore their files consist of a few seconds of bat vocalizations gathered in a controlled setting. **Dataset A** is catered to bat census groups instead, and so contains approximately one minute long files gathered from a recording device being placed in a forest where bats are known to frequent. This results in the files containing many incidental sounds alongside the bat vocalizations, such as wind, insects, and birds.

4.2.1. Dataset A

As a result of needing to record in many locations at once, the recording devices are somewhat lower budget. The audio is recorded at a sampling rate of 250kHz and the resultant files have minimal amounts of filtering. The device was set to record continuously, so much of the data does not contain bat echolocation audio, and other wildlife sounds and microphone static can be heard at most times.

4.2.2. Dataset B

The contents of the files are focused on bat audio, with almost every file containing bat echolocation audio. A high-pass filter is used to exclude sounds below a certain frequency, eliminating most non-bat sounds. The audio of **Dataset B-1** is recorded at a sampling rate of 500kHz, **Dataset B-2** at 375kHz, and for **Dataset B-3** at 250kHz.

4.2.3. PowerShell Script

Automating the process of compressing a large number of files using different algorithms was achieved via the creation of a PowerShell script (available in Appendix B as *Compression Powershell Script*). PowerShell is an open-source program created by Windows that configures and automates task execution using a unique coding language and shell. This script recorded the execution time for each method using Measure-Command, a command that isolates and times specified lines of code. The size of the compressed files was recorded using Measure-Object, which gives the number of bytes contained in one or more files. Table 4.1 below explains the executables used to perform compression, as well as the options used and their meanings.

Executable (version)	**Algorithm / Format**	**Option Used**	**Meaning**
ffmpeg.exe (v 5.0.1)	FLAC MP3	-hide_banner	Do not print copyright notice, build options, or library versions.
		-loglevel error	Only errors will be printed in the log.
MAC.exe (v 7.61)	Monkey's Audio	0	Compression mode (compression level 2000, "normal")
ofr.exe (v 5.100)	OptimFrog	--silent	Suppress log output.
		--encode	Command: encode WAV file(s) to OFR file(s).
tta.exe (v 2.3)	TTA	-e	Encode file.
wavpack.exe (v 5.4.0)	WavPack	-q	quiet (keep console output to a minimum)
7z.exe (v 21.07)	7-Zip (LZMA2) NonSolid 7Zip (LZMA2) Solid 7Zip (PPMd) Solid	a	Add files to archive.
		-t7z	Create a .7z file as output.
		-bb0	Disable output log.
		-ms=off	Disable solid archival mode.
		-ms=on	Enable solid archival mode.

	WavPack7z	-mm=PPMd	Set compression method to PPMd.
		-myx=0	Disable file analysis (so codec selection isn't overridden)
		-m0=WavPack2	Sets WavPack2 as the compression method 0.
		-m1=lzma:d20	Sets LZMA as compression method 1.
		-mb0s1:1	Binds the output of compression method 0 to the input of compression method 1. (thus compressing non-audio data such as headers)

Table 4.1: The executables used to perform compression and the options used.

4.2.4. Execution

The experiment was performed using a single machine possessing an x64-based Intel Core i5-4440 processor, 3.10GHz, and 8GB RAM. It ran on the 64-bit Windows 10 Home operating system, using version 21H1 and the OS build 19043.1645. Care was taken to ensure that all of the machine's processes unrelated to the experiment were either halted or diminished in cases where the process could not be halted. This was done in service of guaranteeing that the measurements of a method's performance were not significantly affected by external factors. Once all methods fully executed their compression of all datasets, the entire process was automatically reiterated until the results of ten iterations were recorded.

4.2.5. Methods Tested

Serving as the independent variable of this experiment, several preexisting audio data compression methods have been tested using the experimental parameters detailed above. The preexisting methods that have had their performance analyzed this way are FLAC, Monkey's Audio, MP3, OptimFrog, TTA, WavPack, 7-ZIP LZMA2 Solid, 7-ZIP LZMA2 NonSolid, 7-ZIP PPMd Solid, WavPack7z, and ZIP.

FLAC

FLAC is a lossless audio codec that supports streaming and seeking the encoded audio. The audio is divided into blocks, and each block is encoded separately, which allows the audio to be quickly decompressed and played starting from any part of the audio [31].

Monkey's Audio
Monkey's Audio is a lossless audio codec that focuses more on good compression ratios rather than quick encoding or decoding while still achieving the necessary speed to support streaming. However, Monkey's Audio is known to have higher system requirements during decoding than usual [8].

MP3
MP3 is a lossy audio codec which supports streaming and seeking. It uses perceptual coding, which permits the codec to discard data or degrade its quality during compression based on a psychoacoustic model [32]. MP3 generally compresses audio to approximately 9% of its original size, though the sampling rate is generally limited to 44.1kHz by the standard, which can cause much stronger reductions in file size if the input audio was of particularly high sampling rate [32]. MP3 was chosen to demonstrate the deleterious effects of lossy compression on ultrasonic data, and is not expected to perform well.

OptimFROG
OptimFROG is a lossless audio codec made with the goal of achieving the best compression ratio possible, largely disregarding the time the process takes. The compressed files do not support streaming or seeking [33].

TTA
TTA (True Audio) is a lossless audio codec that supports streaming and seeking. While its performance is competitive with other lossless codecs [8], TTA has relatively few features [34], which impairs its usability and compatibility and occasionally renders it unable to compress audio files that other codecs can handle.

WavPack
WavPack is a hybrid audio codec, offering both lossy and lossless modes of compression. It supports streaming and seeking, generally focuses more on speed of compression/decompression rather than good compression ratios, and has a relatively wide range of features to enhance compatibility and usability [35].

WavPack7z
WavPack7z is a plugin for 7zip that allows audio data compressed by 7zip to use the WavPack audio codec in place of or alongside its usual techniques [36]. WavPack7z was chosen not for its popularity, but as a comparison to WavPack to explore the possibility that compressing all of the audio files together with an audio codec could be more effective than compressing them individually.

LZMA2 (7zip)
LZMA (Lempel-Ziv-Markov chain algorithm) is a dictionary-based general compression algorithm that features larger dictionary sizes than usual and special support for cases where the distance between matches is consistent [26], [27].

LZMA2 is a wrapper, which is a container format that can contain multiple blocks of data, with each block generally using LZMA compression or remaining uncompressed

[37]. The advantages of this approach are that different LZMA encoding can be used in different blocks, that partially incompressible data can be separated out safely, and that encoding and decoding of multiple blocks can occur simultaneously (e.g. via multithreading).

PPMd (7zip)
PPMd is an implementation of the context-based general compression algorithm Prediction with Partial Match. The implementation used by 7zip is described as "Dmitry Shkarin's PPMdH with small changes" [38]. The general algorithm of PPM, first developed by Cleary and Witten in 1984 [39], predicts each upcoming symbol in the data being compressed using the context established by previously seen symbols in the data. The difference between the predicted next symbol and the actual next symbol is what is stored. In the case of accurate predictions, this significantly decreases the number of different characters that must be stored and thus encoded [26], [27].

DEFLATE (ZIP)
DEFLATE is the general compression algorithm used by ZIP [40]. Deflate first employs LZ77 compression, another dictionary based algorithm, and then Huffman coding, which generates codes for sequences in the data based on their frequency [40].

5. Results

5.1. Interview Responses

5.1.1. Interview A

Interviewee A is a PhD student using ultrasonic recordings to monitor the populations of bats in different habitats. During a project, Interviewee A deploys roughly 20 recorders per night over the course of six or seven weeks. Recording continues uninterrupted for the entire night, approximately 8 hours, generating roughly 10-12GB of data per device per night, which constitutes 2-3TB of data per project.

Interviewee A stores their audio data for an indefinite period of time. From the initial recording device, the data must be transferred multiple times while being processed and backed up. They consider the size of the data to be impactful and storage/transfer to be time-consuming.

"And of course it's really time consuming. Even to just to take them from one place to another is really time consuming."

"Anything to minimize this is really wanted, I think, by anyone who works with ultrasonic calls."

When asked about the relative importance of size, speed, and accuracy in a compression algorithm, Interviewee A said that a loss of accuracy could be acceptable in exchange for a faster compression process and significantly smaller files. This is due to the fact that much of the analysis is only concerned with certain key parameters, and that audio quality outside of these parameters is of lesser importance for species identification. Interviewee A followed up shortly after the interview directing us to a figure from a textbook (Figure 4.2 from *British Bat Calls: A Guide to Species Identification* [18]) describing what these commonly used call parameters are.

During a discussion of the direction of the field, Interviewee A indicated that, to their knowledge, using a multitude of cheap detectors with minimal features and high amounts of noise and empty audio is becoming more popular.

"Wildlife Acoustics have these detectors, actually, you set certain thresholds. So you can say that I only want you to trigger if you really truly think it's a bat and only record what we think are bats, and then you get a lot less noise, but now people are moving to these detectors that I use called AudioMoths. They are super cheap, but they just record everything. That's why you get 10 gigabytes in a night. And so then there's a lot of noise."

5.1.2. Interview B

Interviewee B is a bat biologist doing postdoctoral research on animal behavior and conservation. During a project wherein Interviewee B records bats, they attempt to limit the recording length to approximately 3 seconds. In total, a typical project can generate 1-6TB of data, though long projects can add up to more than that.

Interviewee B considers file storage to be relatively expensive but that it is becoming less of a problem. The generated audio data that is deemed relevant is stored for an indefinite period of time. Long term survey results are usually transferred between several participants once or twice a year. Interviewee B does consider file transfer to be quite problematic.

"Transfer is where the real issue lies, and at the moment the best way to transfer large amounts of audio files is to load them on a hard-drive and physically mail them. This of course is also a time-wasting aspect."

When asked about the relative importance of size, speed, and accuracy in a compression algorithm, Interviewee B responded as follows:

"Accuracy is of utmost importance and completely trumps the other two in my opinion. After that, size would be more important than time for me."

5.1.3. Interview C

Interview C was conducted with a group of professional ecological consultants who have worked on projects requiring large-scale datasets. They provide various ecological services such as surveys and impact assessments, and are often contracted for the purpose of guiding infrastructure development projects. Example projects generated 2-4TB of data each, with individual files varying from 1.5MB to 3MB.

Interviewee C considers data storage and transfer to be problematic. Audio data is generally stored for seven years past the completion of the project. Data is transferred several times as needed for analysis, and is transferred again in full to third-parties for reanalysis from time to time, especially when analysis is challenged on contentious schemes.

When asked about the relative importance of size, speed, and accuracy in a compression algorithm, Interviewee C was adamant that the full, raw audio data needs to be recoverable in full to allow for reanalysis, as mandated by company policy and for some projects by law. After that, the resultant file size is of greater importance than the speed of compression, and that once analysis is over then ease of access to the data is relatively unimportant.

"The presumption to keep raw (unaltered) data may limit the value of any compression software."

"Once analysed, [the data] can sit in the background somewhere; doesn't need to be instantly accessible."

As the recovery of raw audio data is mandated under such strong terms, Interviewee C would require extensive testing to prove that any compressed version of the data would be equivalent to the original, uncompressed WAV files. They also indicated concerns over whether any novel solution would be well-supported enough to not become unavailable in the future.

"Software can rapidly change or becomes unavailable. I think .wav files will be with us for a while, but anything that has a small market share may not, so people with responsibility for storing data may take a little convincing!"

5.1.4. Interviews Summary

- Biologists frequently generate several terabytes of data for each project
- Data storage and transfer is considered onerous due to the size of the data.
- Data is frequently stored for long or indefinite periods of time
- Data is frequently transferred multiple times per project
- Accuracy is of the highest importance, with totally lossless/reversible compression being listed as an absolute requirement by several interviewees.
- Size reduction is considered more important than the speed of compression.

5.2. Experimental Data

Tables 5.1 - 5.4 display the collected data averaged over ten iterations of performing the experiment as described in Section 4.2, with each table showing the results from different datasets. Note that the metric of accuracy is discussed in its own subsection to avoid potentially misleading representations of the data, as any measurement other than 100% was found to both be exceptional and require contextual explanation.

5.2.1. Dataset A

	Speed (seconds)	Compressed Ratio
FLAC	344.4	48.76%
Monkey's Audio	636.6	47.96%
MP3	362.9	1.60%
OptimFrog	1164.0	47.62%
TTA	16.1	0.00%
WavPack	434.9	48.55%
7ZIP LZMA2 NonSolid	11261.8	30.44%
7ZIP LZMA2 Solid	5724.2	30.29%
7Zip (PPMd) Solid	877.8	26.29%
WavPack7z	1907.6	48.44%

ZIP	2299.2	49.50%

Table 5.1: The results produced by compressing **Dataset A**

5.2.2. Dataset B-1

	Speed (seconds)	Compressed Ratio
FLAC	2.9	36.27%
Monkey's Audio	4.7	33.19%
MP3	3.0	1.06%
OptimFrog	9.9	32.09%
TTA	2.5	35.33%
WavPack	3.1	37.06%
7ZIP LZMA2 NonSolid	16.5	40.92%
7ZIP LZMA2 Solid	33.4	41.04%
7Zip (PPMd) Solid	14.0	41.13%
WavPack7z	13.8	34.21%
ZIP	10.2	51.74%

Table 5.2: The results produced by compressing **Dataset B-1**

5.2.3. Dataset B-2

	Speed (seconds)	Compressed Ratio
FLAC	7.8	43.71%
Monkey's Audio	8.4	40.92%
MP3	7.6	0.85%
OptimFrog	19.1	40.15%
TTA	5.2	42.02%

WavPack	6.0	44.24%
7ZIP LZMA2 NonSolid	15.9	46.95%
7ZIP LZMA2 Solid	42.9	46.86%
7Zip (PPMd) Solid	25.5	45.49%
WavPack7z	20.8	41.40%
ZIP	18.0	55.74%

Table 5.3: The results produced by compressing **Dataset B-2**

5.2.4. Dataset B-3

	Speed (seconds)	**Compressed Ratio**
FLAC	11.4	51.50%
Monkey's Audio	21.0	50.64%
MP3	16.3	1.63%
OptimFrog	35.2	50.20%
TTA	11.0	51.71%
WavPack	12.5	50.78%
7ZIP LZMA2 NonSolid	57.2	60.63%
7ZIP LZMA2 Solid	93.1	58.63%
7Zip (PPMd) Solid	115.4	58.24%
WavPack7z	61.6	50.54%
ZIP	35.0	67.66%

Table 5.4: The results produced by compressing **Dataset B-3**

For these tables (see Tables 5.1 - 5.4), speed is expressed as the number of seconds taken by each method to completely compress the given dataset. Size Reduction is expressed as the ratio between the size of the compressed set of files and the size of the initial set of files, converted to a percentage. For example, if given a dataset with a size of 100 bytes, a method with a size reduction of 60% would have compressed the set to a size of 60 bytes on average.

5.3. Accuracy

To measure the accuracy of the algorithms, the compressed files produced in each iteration were uncompressed, and the bytes were compared to the original bytes from the corresponding original file. Only three algorithms were found to have less than 100% accuracy, those being MP3, OptimFrog, and TTA.

As the only lossy algorithm to be tested, MP3 consistently produced files containing none of the original files' critical audio parameters, meaning that the resultant audio could not be in any way used for its intended purposes. The specifics and implications of this will be discussed in later sections, but for the purposes of providing quantitative results, an accuracy of 0% has been given to MP3.

For **Datasets B-1** and **B-2**, the OptimFrog algorithm displayed the following error message for each file that it compressed, where X is the name of the given file:

"**Exception UNKNOWN in file unknown, line 0**
description: Exception READERROR in file unknown, line 0
function ReadFile, code 998, Invalid access to memory location.

Errors occurred compressing <.*X*.WAV>"

In these cases, OptimFrog compressed the files in a way that, when uncompressed, lost 1,600 bytes for each file in **Dataset B-2**, and 16,384 bytes for each file in **Dataset B-1**. All bytes present in the files after compression were also present in their original counterparts, meaning that any loss of accuracy is entirely attributable to these missing bytes. When comparing this data loss to the original file size, this gives OptimFrog an accuracy of 99.31% for **Dataset B-2**, 98.42% for **Dataset B-1**, and 100% for **Datasets A** and **B-3**.

Similarly, TTA was not able to successfully compress any of the data within **Dataset A**, displaying the following error message for each file:

"**tta.exe: can't read from input file**"

Unlike OptimFrog, TTA did not produce any files when encountering this error, and therefore the results presented for TTA (see Table 5.1) should not be interpreted as meaningful performance metrics. Instead, the Speed (seconds) value represents the number of seconds that TTA took to encounter and display the error for all files in the dataset, while the Compression Ratio value indicates that there were 0 bytes present after compression. This outcome gives TTA an accuracy of 0% for **Dataset A**, and 100% for all subsets in **Dataset B**.

6. Analysis

6.1. Experiment

Across all iterations of the experiment for all datasets, the measured metrics were found to be remarkably consistent. For each algorithm with a given dataset, the bytes in the corresponding compressed files had no variation in number or content, confirming their deterministic implementation. To a lesser extent, speed was also found to be rather consistent, as the time taken by a given algorithm to compress a particular dataset rarely fluctuated more than 10%, with most algorithms fluctuating around 5%.

As each dataset is intended to isolate various use cases and contexts to better understand their interactions with the tested algorithms, the results of each dataset are individually analyzed in this section with the intention of highlighting statistical patterns and notable outcomes. Note that due to the significant differences in the amount of data in each set, the measured values for speed vary greatly across different datasets, and as such the bounds of the charts' axis for speed is scaled accordingly. Additionally, all charts in this section represent the compression ratios of each algorithm via a bar graph corresponding to the values of the left y-axis, and the speeds of each via a line graph corresponding to the values of the right y-axis.

6.1.1. Dataset A

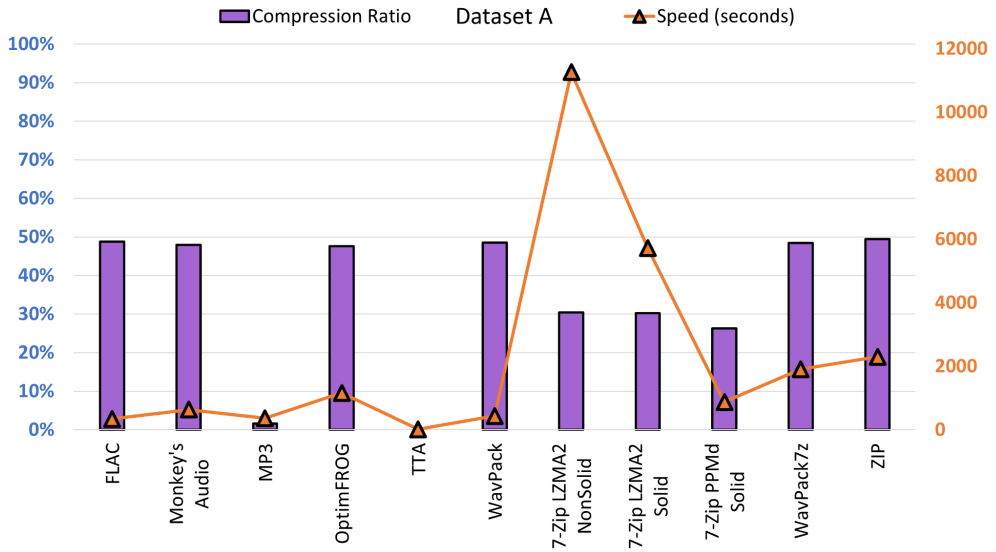


Figure 6.1: The chart visualizing the results produced by compressing **Dataset A**

Dataset A contained the most amount of data across all sets, both in terms of number of files and the size of each file. Within this set were 490 .WAV files totaling approximately 12,850 megabytes, or 12.5 gigabytes. Present within these results are two significant outliers (see Figure 6.1), one being TTA and the other being 7-ZIP LZMA2 NonSolid. As mentioned in a previous section, TTA was not able to successfully compress any of the data within this set, and as such its compression ratio and speed

will not be considered relevant for this analysis. 7-ZIP LZMA2 NonSolid on the other hand was able to successfully compress the data, and is considered an outlier only due to its drastically slower speed. To communicate this exceptional result in a potentially more intuitive way, 7-ZIP LZMA2 NonSolid took almost as long to compress the data as all of the other ten algorithms combined.

FLAC compressed the dataset in the fastest time, averaging a speed of 344.4 seconds. Yielding similar times were MP3 at 362.9 seconds and WavPack at 434.9 seconds. As stated, 7-ZIP LZMA2 NonSolid was found to be the slowest algorithm by far at 11261.8 seconds on average, while the second slowest was 7-ZIP LZMA2 Solid at 5724.2. Notably, out of the three general 7-ZIP algorithms, those being 7-ZIP LZMA2 NonSolid, 7-ZIP LZMA2 Solid, and 7-ZIP PPMd Solid, the third yielded a much faster speed when compared to the other two, with an average of 877.8 seconds and the smallest compression ratio of all the lossless algorithms at 26.29%.

When looking at the variance between the speeds achieved by a given algorithm across all iterations of this dataset, the results for each were remarkably consistent. Taking the ratio between the range of times for an algorithm and the average of those times, the smallest variances were from MP3, OptimFrog, and 7-ZIP PPMd, which ranged from 1% to 2%. The largest variances were measured in WavPack and ZIP, both with around 10.5%, while the others ranged from 4% to 6%. From the perspective of absolute time, however, WavPack and ZIP were quite differently variable, as the former fluctuated between iterations by around 48 seconds, while the latter fluctuated around 4 and a half minutes.

The three general 7-ZIP algorithms were all found to have significantly smaller compression ratios than the other lossless algorithms, the former ranging from 26% to 30% and the latter ranging from 47% to 49%. The only algorithm with a smaller compression ratio than 7-ZIP PPMd Solid was MP3 at 1.60%, which was the only lossy algorithm included in the experiment. As will be seen for all datasets, the compression ratio for MP3 falls far below the other algorithms, consistently yielding ratios approximately 20 to 30 times smaller across all datasets.

6.1.2. Dataset B-1

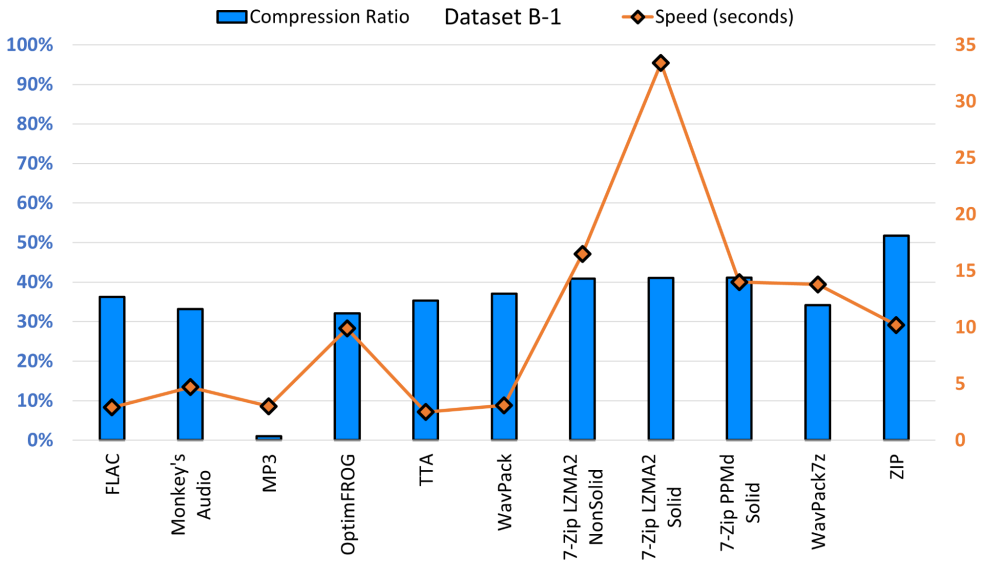


Figure 6.2: The chart visualizing the results produced by compressing **Dataset B-1**

Dataset B-1 was the set with the least amount of initial data, containing 92.2 megabytes over 41 .WAV files, and thus resulted in speeds ranging from approximately 5 to 35 seconds. With the exception of the three general 7-ZIP algorithms, the amount of time taken to compress the complete dataset never had a variance of more than one second. The general 7-ZIP algorithms all produced times with a variance of around two seconds, which when converted to a percentage of the average time, was in a comparable range to the others, that being 2% to 15% across all methods.

The fastest of the algorithms was found to be TTA (see Figure 6.2), with an average speed of 2.5 seconds. FLAC, MP3, and WavPack yielded similar results with speeds of 2.9, 3.0, and 3.1 seconds respectively. A significant outlier in terms of speed was 7ZIP LZMA2 Solid with an average speed of 33.4 seconds, while the next slowest is less than half of that at 16.5 seconds.

In terms of size reduction, MP3 yielded a compression ratio of 1.06%, while the next lowest ratio was OptimFrog's 32.09%, followed by Monkey's Audio with 33.19%. ZIP yielded the least amount of compression, achieving a compression ratio of 51.74%, and the remaining algorithms fell in the range of 34% to 41%. With the exception of the highest and lowest ratios appearing to act somewhat as outliers and the three general 7-ZIP algorithms all nearly the same at approximately 40%, the resultant ratios fall evenly spread across the 34% to 37% range.

6.1.3. Dataset B-2

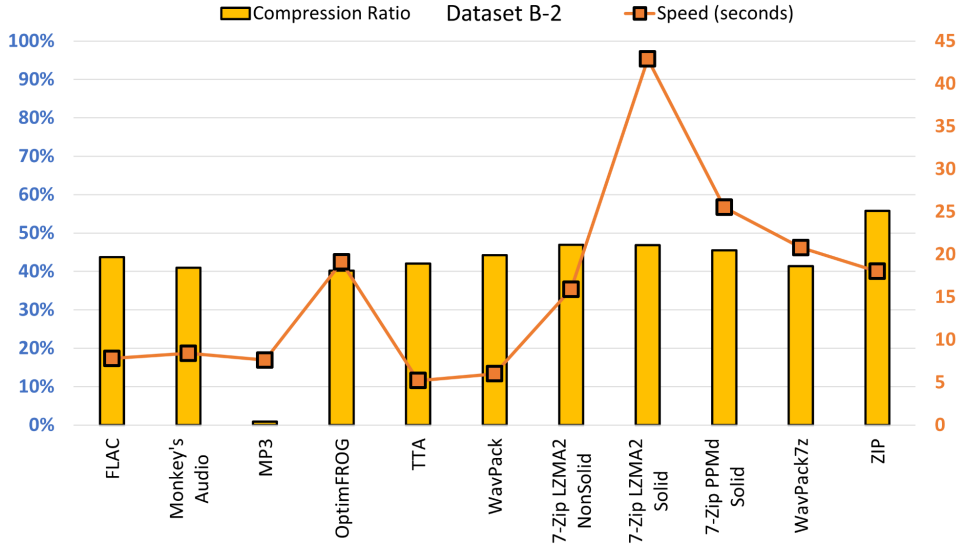


Figure 6.3: The chart visualizing the results produced by compressing **Dataset B-2**

As can be seen by comparing Figure 6.2 and Figure 6.3, the results of compressing the **Dataset B-2** followed a similar pattern to the results of **Dataset B-1**. Just as in that dataset as well as **Dataset A**, the metric of size reduction was consistent for each algorithm down to the byte, and the speeds of most algorithms varied less than a second across all iterations. When looking at the variation in speed compared to the corresponding average time, all but two methods possessed a variation of 5% or less, those expectations being ZIP and 7-Zip LZMA2 NonSolid. These algorithms had a variation of 10% and 15% respectively, and as the amount of data in this set is approximately 50% larger than **Dataset B-1**, containing 138 megabytes over 145 files, the consistencies between these times are statistically more significant.

TTA was again found to be the fastest algorithm with a speed of 5.2 seconds, with WavPack, MP3, and FLAC close behind with 6.0, 7.6, and 7.8 seconds respectively. Additionally, 7ZIP LZMA2 Solid was exceptionally slow with a speed of 42.9 seconds, however the next slowest speed was 25.5 seconds, meaning that 7ZIP LZMA2 Solid did not deviate as far from the others as in **Dataset B-1**.

For all but one algorithm, size reduction was measured to be larger in comparison to those measured in **Dataset B-1**. They were more closely spread, mostly ranging from 40% to 47%, with the three general 7-ZIP algorithms occupying the higher end of that spectrum, and ZIP yielding a significantly larger ratio of 55%. The previously mentioned algorithm that is exceptional to this is MP3, which yielded the smallest compression ratio of 0.85%, slightly smaller than what it was measured as in **Dataset B-1**. OptimFrog and Monkey's Audio followed behind with their respective compression ratios of 40.15% and 40.92%, continuing to mirror the results of **Dataset B-1**.

The heavy similarity between the results of these two datasets is indicative of, and likely caused by, the similarities in the content and context of their audio recordings. As previously stated, these datasets contain around a second or two of isolated bat vocalizations, and were separated based on their sample rate.

6.1.4. Dataset B-3

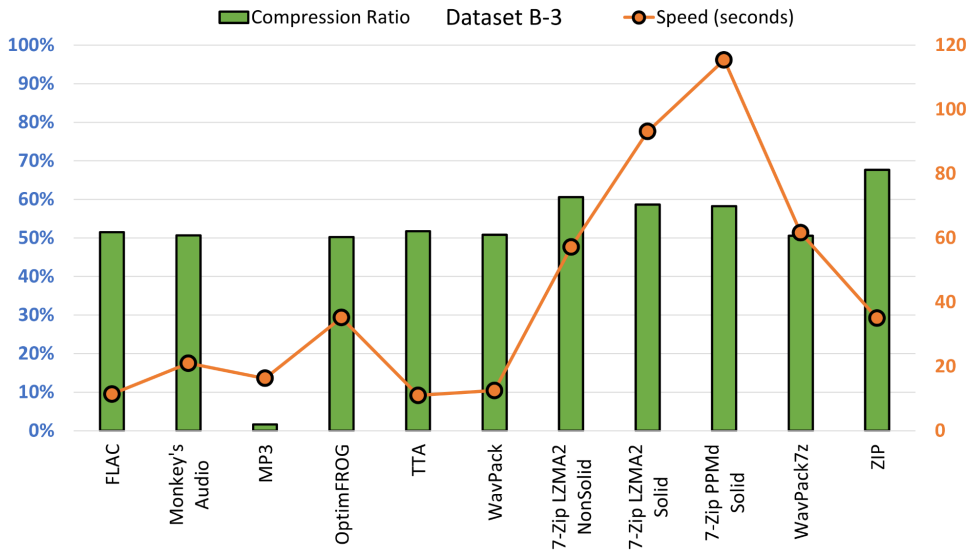


Figure 6.4: The chart visualizing the results produced by compressing **Dataset B-3**

Among all the subsets of **Dataset B**, **B-3** had the most data at 377 megabytes over 185 files, although this is still much less than **Dataset A**. Just as in all other datasets, there was no variance in the number or content of bytes measured in the compressed files for each algorithm across all iterations, and a slight variance in compression speed. 7-ZIP LZMA2 NonSolid was found to have the largest variance in speed with 7.5%, which is approximately half the largest variance measured for any one algorithm measured in both **B-1** and **B-2**.

While TTA, WavPack, and MP3 were once again the fastest performing algorithms (see Figure 6.4), the slowest speed for this dataset was found to be 7-ZIP PPMd Solid with a speed of 115.4 seconds.

Size reduction across all algorithms was significantly larger than both **B-1** and **B-2**. Moreover, the results followed a similar pattern where ZIP yielded the highest ratio, followed by the three general 7-ZIP algorithms producing comparable results to each other, then followed by the remaining lossless algorithms being evenly distributed across a few percentage points of each other, and MP3 being orders of magnitude lower. In this case, the distribution of the smallest lossless algorithms were much closer together than in **B-1** and **B-2**, where six algorithms had ratios that fell within a range of 1.5 percentage points.

6.1.5. Dataset B Overall

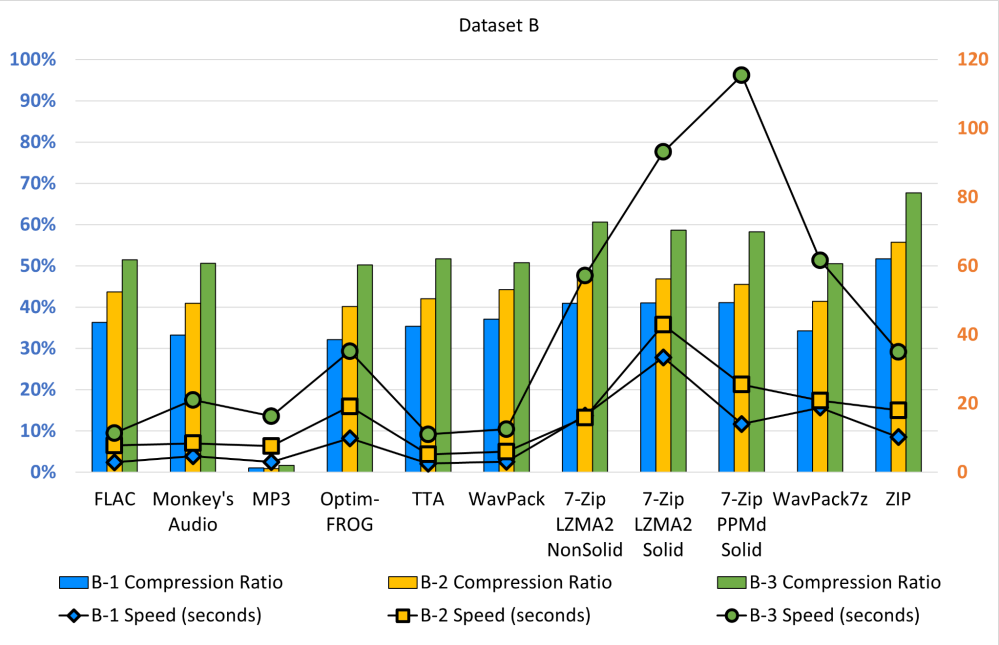


Figure 6.5: The chart visualizing the aggregated results from **Dataset B**

Analyzing the behavior of the metrics across all subsets of **Dataset B** shows a consistent relation between the performance of the algorithms (see Figure 6.5). MP3 had the smallest compression ratio to a strong extent, OptimFrog had the smallest of any lossless algorithm, although many others were quite close, and ZIP had the largest. TTA achieved the fastest compression time while the slowest was typically 7-ZIP LZMA2 Solid, with a single outlier in **Dataset B-3**, where 7-ZIP PPMd Solid was the slowest.

What can also be seen in Figure 6.5 is that every algorithm yielded the smallest compression ratio in **Dataset B-1** and the largest in **B-3**, with the single exception of MP3. Speed fits this pattern as well, however this can be attributed to the number of bytes per dataset increasing, as **B-1** has the least number of bytes and **B-3** has the most, and so this correlation is not meaningfully insightful. The compression ratio on the other hand is a percentage of the initial number of bytes, suggesting that there may be some other contributing factor or impactful emergent property.

6.1.6. General Compression Algorithms

Of particular note was the performance of general compression techniques, such as LZMA2 and PPMd (via 7zip) and DEFLATE (via Zip), on **Dataset A**. Most of the compression techniques tested maintained fairly consistent competitiveness across the different datasets, for instance Monkey's Audio consistently achieving a slightly better compression ratio than FLAC at the cost of taking almost twice as long. However, the general compression algorithms performed remarkably better than audio codecs, but only for **Dataset A**, compressing files to an average size of 30.37% for LZMA2 Solid and 26.29% for PPMd Solid. It is quite unusual for a focused compression algorithm to perform worse than a general one, so further investigation was performed.

Firstly, the audio files were listened to and viewed as spectrograms. This revealed a significant amount of static and low-intensity noise in the files of **Dataset A**, and relatively low amounts of distinct or high-intensity sounds. Intuitively this makes sense, as **Dataset A** came from lower-quality recorders which are left on for long periods of time, meaning that the recording quality would be lower and there would be fewer periods of high activity proportionally. A comparison between the spectrograms of example files from each dataset is shown in Figure 6.6 as a demonstration.

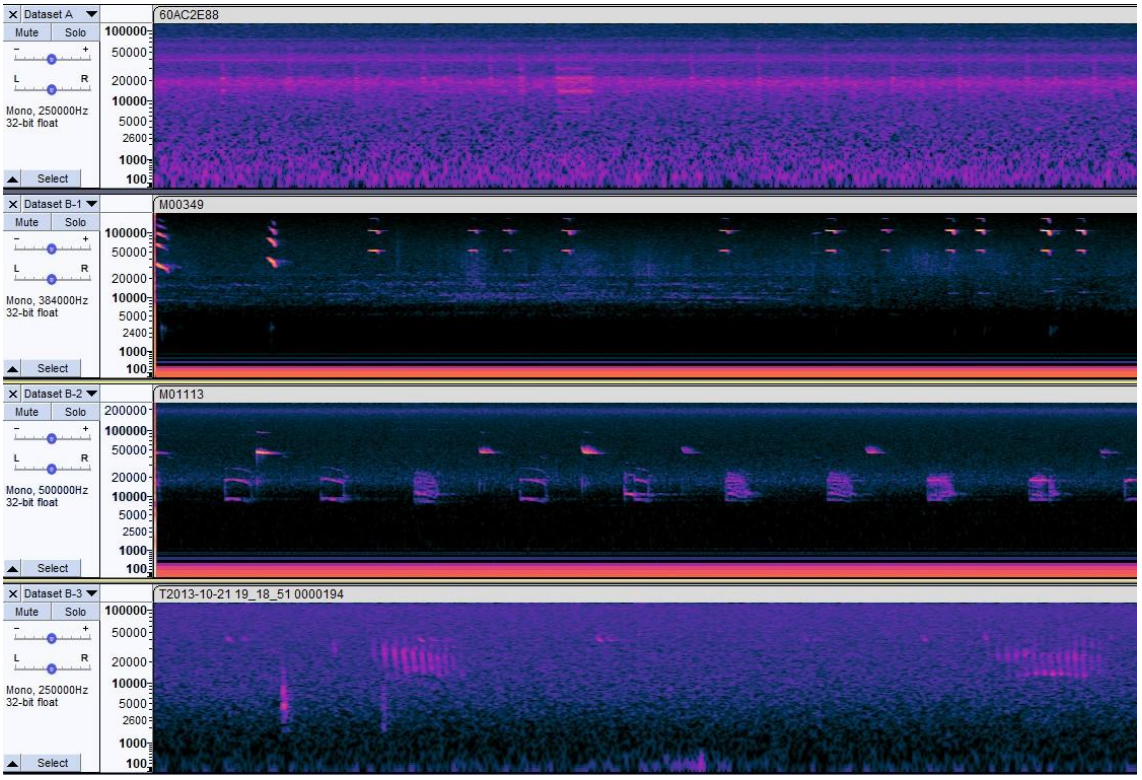


Figure 6.6: Example spectrograms from **Datasets A** (top), **B-1**, **B-2**, and **B-3** (bottom).

In an attempt to measure the amount of static noise versus high-intensity sounds in these files, a Python script was written (available in Appendix B as *Sample Analysis Python Script*) to convert the audio to arrays of samples and calculate the ratio of the following two values: the difference between the maximum and minimum sample (range), and the standard deviation of the samples (σ). The hope was that the amount of noise in a file would be correlated with either the effectiveness of the general compression algorithms or the relative ineffectiveness of the audio codecs. While the results seem to suggest that this calculation properly captures the noise levels visible in the spectrogram, this does not appear to relate to the difference in compression efficiency. This is primarily due to the poor effectiveness of general compression techniques on **Dataset B-3**, despite it also featuring static noise. See Figure 6.7 below for example raw outputs from the script and Table 6.1 for the average range/σ values alongside some example compression ratios each dataset achieved.

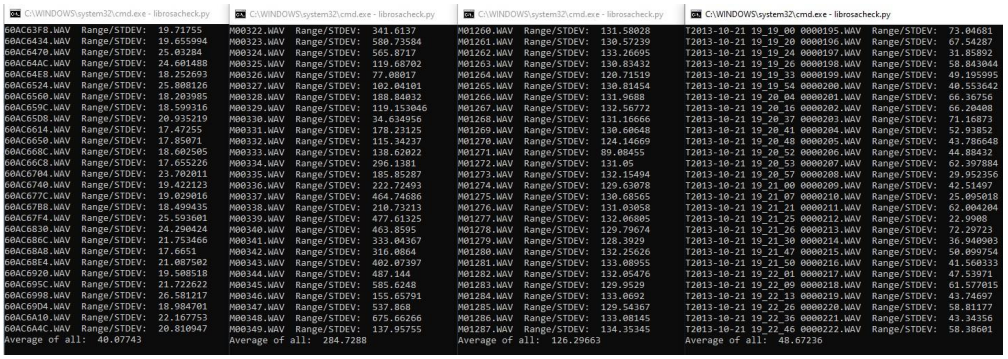


Figure 6.7: Raw outputs from the Sample Analysis Script.

	Dataset A	Dataset B-1	Dataset B-2	Dataset B-3
Range / σ	40.08	284.73	126.30	48.67
Compression Ratio (FLAC)	48.76%	36.27%	43.71%	51.50%
Compression Ratio (PPMd)	26.29%	41.13%	45.49%	58.24%

Table 6.1: The average Range/Standard Deviation of the samples of each dataset, compared to two example compression ratios they achieved.

6.1.7. Overview

Across all datasets, some commonalities can be found. In terms of speed, the slowest was always one of the general 7-ZIP algorithms, although none of them can be thought of as the slowest since the dataset dramatically influenced their relative rankings. The same cannot be said about compression ratio, however, as ZIP yielded the largest ratio in every dataset by a significant margin and MP3 yielded the smallest. When considering only the lossless algorithms, **Dataset B** consistently found the smallest compression ratios in OptimFrog, Monkey's Audio, and WavPack7z, all of which being within only one or two percentage points of each other. **Dataset A** does not share this pattern however, as the general 7-ZIP algorithms yielded significantly smaller ratios, with 7-ZIP PPMd Solid achieving the smallest ratio of all lossless algorithms across any dataset.

7. Discussion

7.1. Interviews and Desiderata

In order to address SQ1: *What are the desiderata for an audio compression method from the perspective of biologists specializing in ultrasonic frequencies?* interviews were conducted with a number of experts working with ultrasonic bat audio.

Interviewees were chosen from a range of fields relevant to ultrasonic bat echolocation audio, and interviews conducted with biocensus consultants working in industry, a field researcher, and a lab researcher.

All of the interviewees indicated that they store data for long or indefinite periods of time, transfer entire datasets multiple times per project, and generally consider the size of the data to be onerous.

Two of the interviewees indicated a strong need for lossless compression, one for scientific reasons and one for legal/contractual reasons. Moreover, all used programs for analysis which require the use of raw WAV files. Consequently, any compression method would need to be reversible, potentially multiple times without degradation, ruling out the possibility of lossy audio compression techniques even for the interviewee who was amenable to accuracy loss.

All of the interviewees indicated that file sizes were of importance to them, and expressed notable irritation with the lack of available options for managing the magnitude of data.

None of the interviewees indicated any use cases that demanded a degree of speed for compression or decompression anywhere near what most popular audio codecs boast. This is due to the fact that analysis must be done on WAV files, so compression is only concerned with easing storage and transfer, in contrast with audio codecs which aim for additional features such as being immediately playable and seekable.

Overall, the desiderata of an audio compression method, from the perspective of biologists specializing in ultrasonic frequencies, imply the following qualities in a solution: a lossless compression technique that emphasizes compression ratio but does not entirely sacrifice compression speed. A proper audio codec is not necessary, as the compressed files do not need to be playable or seekable. The ability to quickly retrieve a single file from storage is not necessary, but the data tends to be split into a large number of very small, similar files, which implies solid compression techniques could be effective.

7.2. Experiment

To address SQ2, *How well are these desiderata satisfied by currently available methods?*, as well as SQ3, *Which of the examined methods can be recommended for use by the target group within the various contexts they operate in?*, a controlled experiment was performed judging the effectiveness of various compression techniques on data provided by experts working with ultrasonic bat audio.

7.2.1. Monkey's Audio

Out of the lossless audio codecs, Monkey's Audio displayed the best compression ratio without suffering issues with compatibility or file destruction. However, Monkey's

Audio also took substantially longer to compress the data than the main competing codecs, FLAC and WavPack. Moreover, Monkey's Audio is known to consume more system resources than usual, and on certain devices this may become prohibitive. This would especially be the case if it were desirable that compression occur on the recording device. Doing so would allow the devices to record for longer, but lower-quality devices may not have the additional processing power required to support a heavy compression process as offered by Monkey's Audio.

According to the desiderata established by the interviews, Monkey's Audio qualifies it to be the main recommendation, with the exception of using 7zip PPMd on data resembling **Dataset A** (more on that in the *General Compression vs Audio Codecs* subsection). However, the additional demands may make Monkey's Audio infeasible for certain use cases, so it cannot be the sole recommendation.

7.2.2. FLAC and WavPack

Both FLAC and WavPack consistently performed quite well across all of the datasets. As expected from lossless codecs, none of the audio data was compromised upon compression and decompression. Both codecs achieved good compression ratios and very quick compression speeds. These codecs can be equally recommended for data similar to **Datasets B-1**, **B-2**, and **B-3**, in cases where the additional resource requirements for Monkey's Audio cannot be met, when compression time is of higher value than the small gain in compression efficacy, or when a popularized compression method is needed as indicated by Interviewee C.

It was noted during testing that the decompression rate of FLAC seemed slightly superior to WavPack, but as this was not mentioned as an important desideratum by interviewees it was not included as a controlled measure in the experiment and should not be used to form a conclusion.

As the two came out so similarly, neither one can be recommended over the other based on the primary desiderata. Ancillary features such as compatibility with certain programs or hardware, tagging preferences, or WavPack's support for a lossy algorithm under the same format may be the deciding factor between these two.

7.2.3. MP3

MP3 was found to have the greatest size reduction, consistently achieving a compression ratio slightly above 1% of the original file size. It also performed the task fairly quickly, achieving a speed comparable to the fastest algorithms for each dataset. However, MP3's accuracy was by far the worst of the algorithms tested, making it not viable for the use cases examined in this thesis. This accuracy loss is partly due to the generally lower fidelity that lossy compression methods aim for, as well as the psychoacoustic model of MP3 removing sounds outside of the range of human hearing.

As an example, see audio file M01250 from **Dataset B-2** in Figure 7.1 below, expressed as a Mel spectrogram of the original WAV file (top) and the compressed MP3 file (bottom). Take special note of the difference in scales, as the MP3 is limited to 24kHz frequencies, destroying virtually all of the useful data from the ultrasonic audio in the WAV.

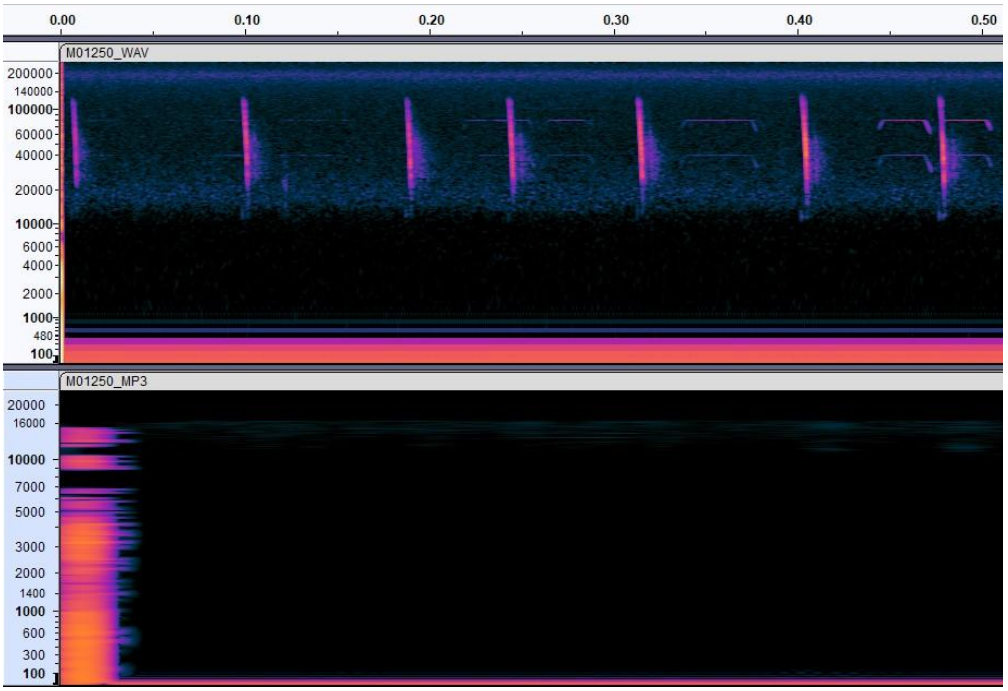


Figure 7.1: Example of a WAV file and the loss of ultrasonic data in the resulting MP3.

This destruction cannot be trivially remedied, as it's based on the limited bit rate/sampling rate. As per the Nyquist-Shannon sampling theorem, the highest pitches that can be recorded are of frequencies less than half the sampling rate. In order to record the sounds typical for bat echolocation research, a sampling rate of at least 250kHz would be necessary, which would multiply the size of the compressed file by a factor of approximately 5 compared to its current 48kHz sampling rate. Given that MP3 is generally expected to achieve a compression ratio of 9%, increasing the sampling rate to 250kHz would improve the quality of the audio, but at that point a ~45% compression ratio would yield very little advantage over true lossless compression.

7.2.4. Optimfrog

As mentioned in Section 5.3, the OptimFrog algorithm consistently produced an error message when compressing files in **Datasets B-1** and **B-2**. To better understand what had occurred to produce the error message, the compressed files were decompressed and analyzed. As OptimFrog is a lossless algorithm, the bytes present in the decompressed file should be identical to the original corresponding file. When comparing the bytes of the original file and decompressed file, it was found that all bytes of the files were identical until the end, at which point the decompressed file ended sooner than the original. For **Dataset B-2**, all decompressed files were found to be missing exactly 1,600 bytes from the end of the original file, and all decompressed files from **Dataset B-1** were missing exactly 16,384 bytes each.

Further inspection found that while the standard .wav format contains a header with a length of 44 bytes, the files in **Dataset B-1** contained 1024 bytes headers and **Dataset B-2** files had 288 byte headers. Decoding these extra bytes into readable text showed

that extra headers were added to the files by the recording device and its software. While the other tested methods were able to correctly identify this additional header data and begin compression at the correct byte offset, OptimFrog assumed a header of exactly 44 bytes, and was therefore unable to begin and end the compression process at the correct positions.

Also of note is that while these shorter uncompressed files are still able to be listened to with this ending data removed, OptimFrog cannot cannot compress the file a second time. Attempting to do so produces an **UNEXPECTEDEOF** error message, indicating that the ending bytes that were lost in the initial compression were necessary for OptimFrog to successful compress the file.

7.2.5. TTA

Unlike OptimFrog, TTA's failure to compress **Dataset A** was not found to be attributable to any misreading of header data. Modifying the bytes present in the headers of these files so that their structure more closely resembled the other sets produced the same error. While the exact mechanism that led to this failure was not able to be determined in the scope of this paper, this behavior is seemingly consistent with the reputation that TTA has of lacking the compatibility of other codecs.

7.2.6. WavPack7z

WavPack7z was included to explore the possibility that compressing all of the audio files together with an audio codec could be more effective than compressing them individually. It appears that using WavPack7z generally compresses slightly more efficiently than WavPack on its own, at the cost of taking substantially longer. However, the compression efficiency of WavPack7z did not manage to beat the efficiency of other audio codecs focused on efficiency such as Monkey's Audio, and so for the use case being examined in this study does not seem to be a worthwhile option.

7.2.7. General Compression vs Audio Codecs

Broadly speaking, the general compression algorithms performed more slowly and achieved worse size reduction than the dedicated audio codecs–with one notable exception in Dataset A–though both types of algorithms managed to maintain 100% accuracy. While this result was expected, the use of compression algorithms that do not specifically focus on the compression of audio was important in order to gauge the relative effectiveness of the techniques which do, as this gives insight into the effectiveness of this focus. Overall, it seems that audio codecs benefitted less in terms of compression ratio when used on the ecological data in this thesis than they often do when compressing human-focused audio such as speech and music, though they did still benefit. This implies either that some of the unique qualities of the ecological audio are not being considered or that design choices made in consideration of the unique qualities of speech and music are not applicable in this ecological audio.

As mentioned in the analysis, it was found that general compression algorithms were exceptionally effective on Dataset A, whereas audio codecs maintained roughly similar effectiveness across all the datasets. Investigating the files more closely revealed a noticeably higher amount of static noise relative to activity spikes in the files of Dataset

A. However, analysis was not able to confirm that this was related to the unusual effectiveness of general compression techniques. This is because Dataset B-3 also featured significant, albeit lesser, levels of static noise relative to activity spikes, but was far less effectively compressed by the general compression algorithms in comparison to the audio codecs.

While it is possible that either **Dataset A** or **Dataset B-3**'s performance was anomalous, and that noise versus activity is indeed related to general compression algorithm effectiveness, it cannot be seen from only the available data which is anomalous. It is therefore not possible to conclude whether this metric is of value, whether there are other metrics that could reveal the surprising effectiveness of general compression algorithms, or whether some of the data is damaged or otherwise unrepresentative of ordinary ultrasonic bat audio data.

Though it cannot be proven from this experiment whether the solution will apply generally to data gathered in the same style as **Dataset A**, within the scope of this study the recommendation is to use 7zip's PPMd algorithm to compress such data due to its outstanding size reduction while maintaining 100% accuracy.

7.2.8. Solid vs Nonsolid

In this experiment, the LZMA2 algorithm was tested twice, once creating a solid archive and once a nonsolid archive, in order to test the difference between the two. Broadly speaking, the results were as theory would predict in that it took substantially more time to create a solid archive than a nonsolid one, but the compression efficiency of the solid archive was superior.

Somewhat surprising, however, was how small the difference in efficiency was. Solid archival generally grants the greatest improvements when the input is a large number of small, similar files, which would have implied more significant improvements in the datasets tested than what was experimentally tested. As it stands, it does not seem like solid archival of audio files is worth the added time and risk of data corruption that it costs, though more work could be done in the future to see if other algorithms gain more efficiency from being made solid rather than not.

7.2.9. Datasets

As previously discussed, the files used in the experiment were categorized into datasets based primarily on the context of their recording, such as date, location, and sample rate. Thus, the amount of data in each dataset differs from each other quite drastically, ranging from 92 MB in **Dataset B-1** to 12,850 MB in **Dataset A**, as some contexts produced more data than others. While this can partially be attributed to recordings with higher sample rates requiring more data per second of audio, the number and duration of files varies for each set vaguely proportionally. For example, **Dataset A** contained 490 files with an average duration of 55 seconds, while **Dataset B-1** had 41 files at an average of 3 seconds. This means that the results corresponding to larger datasets are more likely to be statistically significant in comparison to smaller datasets, as those values represent the average performance for a given algorithm with respect to that dataset. Phrased in a potentially more intuitive way, having more data to compress requires more execution time for an algorithm, which in turn provides more

measurements of that performance, thus reducing the impact that any exceptional measurement has on the final average.

While files may have been omitted from sets in an attempt to have all datasets contain a consistent number of files, bytes, or seconds of audio, the difference in sample rates means that it is not possible to have all of these metrics consistent. Additionally, these sets were intended to represent use cases common to the target group, and the total amount of data and number of files produced cannot be thought of as independent from that context. While it is possible that the performance of one or more algorithms is meaningfully correlated to the number of files within a set or the average duration of those files, the execution of other experiments that are designed to isolate these variables would be required in order to confirm or deny this.

8. Conclusions and Future Work

8.1. Research Questions

In Section 1.3, *Problem Formulation*, four research questions were presented as a framework for the structuring and interpretation of this paper and the work done within it, one main question and three sub-questions. This section presents the knowledge and findings of the project as detailed in previous sections as answers to these questions.

SQ1. What are the desiderata for an audio compression method from the perspective of biologists specializing in ultrasonic frequencies?

The desiderata mentioned in SQ1 were elicited primarily from the data collected during interviews, while following the precedent set by previous related works [3], [8], [9], [10]. These studies served as a foundation for how to objectively measure and compare the contextual value of audio data compression algorithms, and provided a standardized means of communicating these metrics to the target group. The metrics of data accuracy, file size reduction, and compression speed were selected from and defined by these works, which were then presented to the target group in interviews to judge their applicability and sufficiency.

To summarize the elicitation within Section 7.1, *Interviews and Desiderata*, the responses generated from the interviews strongly suggest that the three metrics of accuracy, size reduction, and speed are the main desiderata for an audio compression method within the context of bat biology. Moreover, with some degree of contextual variance, the metrics are prioritized as accuracy being the highest, and speed being a tertiary yet still significant concern.

SQ2. How well are these desiderata satisfied by currently available methods?

With a thorough understanding of what the target group expects and demands from an audio data compression algorithm, the experimental results may now be interpreted in a more qualitative manner. Generally speaking, the algorithms analyzed in this paper fall short of being a versatile tool for all or even most bioacousticians. The degree to which they satisfy the established desiderata varies significantly depending on the specific properties of the files' content, with some algorithms failing to compress files in certain contexts entirely. As such, the value of each algorithm must be thought of as contingent on whether they are to be considered advisable for legitimate use. Moreover, as the properties that cause this contingency are either obscure or ambiguous, there lacks precision in the ability to identify use cases where the performance of an algorithm can be expected to be worthwhile.

With this in mind, the tested algorithms predominantly do not sufficiently satisfy the desiderata established, even when properly contextualized. While most algorithms did achieve 100% data accuracy, their ability to reduce the size of a file does not reach a point that would materially affect bat biologists' ease in storing or transmitting data. This would likely not change the feeling that physically mailing hard-drives is the best way to transfer large amounts of audio data. So, while the existing recommendations do offer value, there remain significant challenges in managing so much data as well as clear avenues for improved processing of this specific type of audio data.

SQ3. Which of the examined methods can be recommended for use by the target group within the various contexts they operate in?

As discussed in the previous sections, the algorithm with the best compression ratio across all datasets was MP3, which also achieved speeds that were near what the fastest algorithms were measured at. However, as these metrics were strictly the byproduct of MP3's drastically low accuracy, it cannot be considered a viable solution in the context of ultrasonic bioacoustics.

Similarly, as both TTA and OptimFrog were not able to accurately parse files created by software commonly used in the field, they too should be discarded as valid options. Even though TTA and OptimFrog achieved 100% accuracy for at least half of the datasets, the reasons that lead to the files being corrupted cannot be easily predicted by those with only a casual knowledge of data compression, as this paper assumes of the target group. Furthermore, when compared against alternatives within datasets where these algorithms did not produce any data loss or corruption, their performance in all metrics were not found to be significantly better, meaning that their risk of failure is not meaningfully outweighed by their successes.

Therefore, the remaining algorithms which have consistently compressed all datasets without any loss of data will be considered equal in the metric of accuracy, as all of them yielded a result of 100% for all iterations with no measured variance. Comparing these algorithms will focus primarily on the other two metrics of speed and size reduction, with the latter being prioritized higher in accordance with the results of the case studies.

Dataset A was intended to represent use cases where lower cost recording devices were placed in the wild and set to record mostly continuously, as opposed to a more controlled laboratory environment. In this case, 7-ZIP PPMd Solid was found to achieve remarkably good size reduction at almost a quarter of the original size, while still yielding reasonable speed and perfect accuracy. However, the extent to which this result is generalizable is unknown, as more testing is needed to identify the reasons behind this outcome and why similar performance is not measured in other datasets. For these reasons, this algorithm is recommended to be used over the alternatives in this context, with a note that care should be taken to ensure that the data intended to be compressed is sufficiently similar to the test data. Should speed ever outweigh size reduction as a priority in these cases, especially if the ability to stream the audio is needed, FLAC and WavPack consistently perform approximately 2.5 times faster, although their size reduction is much worse, yielding around half of the original size.

Dataset B was split into three subsets based on their initial sampling rate and recording conditions, although all were intended to represent the use cases of short files containing isolated bat audio recordings. While interesting patterns arise when looking at the metrics and how they change with respect to the sample rate, the conclusion of which algorithm to recommend in practical use is relatively consistent throughout. Monkey's Audio yields the best size reduction, although its competitors are not as far behind in comparison to **Dataset A**. For example, in **Dataset B-3** Monkey's Audio achieves a size reduction less than 1 percentage point better than FLAC and WavPack, though FLAC and WavPack perform almost twice as fast. With this in mind, Monkey's

Audio is recommended for this context in spite of this small difference, as an improvement of 1 to 3 percentage points can be very impactful when on the scale of several terabytes. However, if Monkey's Audio is too resource intensive for a particular usage, for instance if compression is to be performed on the recording device and computer power is very limited as a result, FLAC and WavPack serve as good alternatives.

Finally, results seem to indicate that solid compression is not particularly worthwhile for any of the datasets, given the risks it imposes and the extra time it takes in exchange for relatively little compression efficiency. However, solid vs nonsolid archival was only formally tested for LZMA2, so there remains a chance that nonsolid compression would be significantly more detrimental when using other techniques.

What are the best performing audio data compression methods in the context of ultrasonic bat bioacoustics?

With the sub-questions answered, the answer to the main question can be properly articulated. Due to the high demand of perfect accuracy from the target group, lossless algorithms can be considered better than lossy, especially when accounting for the significant bias against ultrasonic frequencies present in most lossy algorithms. Between lossless algorithms, FLAC, WavPack, and Monkey's Audio all consistently perform better than their competitors, although the unique results of **Dataset A** suggest that 7-Zip PPMd Solid is the best algorithm within specific contexts.

8.2. Future Work

There exists a possibility that a novel compression algorithm could be designed with the specific intention of catering to desiderata specific to the context of ultrasonic bioacoustics. Due to the high complexity of implementing an algorithm of that scale from scratch, such a project would most likely begin as an extension of or a modification to an existing compression algorithm. As such, this paper may serve as foundational literature that assists in the selection of an algorithm to work with, and may provide a deeper understanding to the mechanisms that impact its eventual design and implementation.

Furthermore, the conclusion of the experiment performed in this paper suggests a significant difference between the two use cases represented by the datasets. As previously mentioned, 7-ZIP PPMd Solid yielded the best size reduction in **Dataset A**, but was consistently much worse in **Dataset B**. A potential direction to extend this work would be to create a means of identifying which use case a given audio file falls under, and therefore which algorithm would produce the best results for that file. This would not only assist those in the target group by providing further information relevant to their context, but it would quite likely deepen the understanding of how and why these algorithms produce their respective results, providing further foundational literature. Moreover, PPMd's effectiveness on **Dataset A** granted the best compression ratios out of the entire experiment, and Interviewee A's claim *"now people are moving to these detectors that I use"* implies that this type of file may become more popular in the near future. Therefore, a compression algorithm tailored to these low-quality recordings with long periods of low activity could be of substantial use to the field of bat research.

Finally, more research remains to be done on whether these results would apply to other bioacoustic purposes, for instance in the recording of other animals or environments. While recording ultrasonic frequencies causes especially outsized difficulties for bat researchers, the general concept of gathering long recordings with long periods of silence on lower-quality recorders could also apply to other wildlife researchers. As progress is made in developing solutions for bat researchers, more studies on the generalizability of these findings could bring benefits to adjacent fields.

References

[1] A. Dobson, “Titley Scientific FAQs,” *Titley Scientific*, Jun. 19, 2010. https://www.titley-scientific.com/eu/support/faqs (accessed Feb. 02, 2022).

[2] T. Painter and A. Spanias, “Perceptual coding of digital audio,” *Proceedings of the IEEE*, vol. 88, no. 4, pp. 451–515, Apr. 2000, doi: 10.1109/5.842996.

[3] B. E. Heath, S. S. Sethi, C. D. L. Orme, R. M. Ewers, and L. Picinali, “How index selection, compression, and recording schedule impact the description of ecological soundscapes,” *Ecology and Evolution*, vol. 11, no. 19, pp. 13206–13217, Aug. 2021, doi: 10.1002/ece3.8042.

[4] M. P. Mcloughlin, R. Stewart, and A. G. McElligott, “Automated bioacoustics: methods in ecology and conservation and their potential for animal welfare monitoring,” *Journal of The Royal Society Interface*, vol. 16, no. 155, p. 20190225, Jun. 2019, doi: 10.1098/rsif.2019.0225.

[5] A. López‐Baucells, N. Yoh, R. Rocha, P. E. D. Bobrowiec, J. M. Palmeirim, and C. F. J. Meyer, “Optimizing bat bioacoustic surveys in human‐modified Neotropical landscapes,” *Ecological Applications*, vol. 31, no. 6, Jun. 2021, doi: 10.1002/eap.2366.

[6] J. Xie, J. G. Colonna, and J. Zhang, “Bioacoustic signal denoising: a review,” *Artificial Intelligence Review*, vol. 54, no. 5, pp. 3575–3597, Nov. 2020, doi: 10.1007/s10462-020-09932-4.

[7] R. C. Whytock and J. Christie, “Solo: an open source, customizable and inexpensive audio recorder for bioacoustic research,” *Methods in Ecology and Evolution*, vol. 8, no. 3, pp. 308–312, Nov. 2016, doi: 10.1111/2041-210x.12678.

[8] M. van Beurden, *Lossless Audio Codec Comparison*, 4th ed. 2015. [Online]. Available: http://www.audiograaf.nl/losslesstest/Lossless%20audio%20codec%20comparison%20-%20revision%204.pdf

[9] T. Hidayat, M. H. Zakaria, and N. Che Pee, “Comparison of Lossless Compression Schemes for WAV Audio Data 16-Bit Between Huffman and Coding Arithmetic,” *International journal of simulation: systems, science & technology*, Feb. 2019, doi: 10.5013/ijssst.a.19.06.36.

[10] H. Chen and T. L. Yu, “Comparison of Psychoacoustic Principles and Genetic Algorithms in Audio Compression.” Accessed: May 16, 2022. [Online]. Available: http://dx.doi.org/10.1109/icseng.2005.28

[11] B. J. Cardinale *et al.*, “Biodiversity loss and its impact on humanity,” *Nature*, vol. 486, no. 7401, pp. 59–67, Jun. 2012, doi: 10.1038/nature11148.

[12] J. Joshi, S. Porwal, Y. Chaudhary, and M. Jain, “Data Compression Methodologies for Lossless Data and Comparison between Algorithms,” *International Journal of Engineering Science and Innovative Technology (IJESIT)*, vol. 2, no. 2, pp. 142–146, Mar. 2013.

[13] U. Flick, *The SAGE Handbook of Qualitative Data Analysis*. SAGE, 2013.

[14] J. W. Knopf, “Doing a Literature Review,” *PS: Political Science & Politics*, vol. 39, no. 01, pp. 127–132, Jan. 2006, doi: 10.1017/s1049096506060264.

[15] J. Blair, R. F. Czaja, and E. A. Blair, *Designing Surveys*. SAGE, 2013.

[16] S. Messick, “VALIDITY,” *ETS Research Report Series*, vol. 1987, no. 2, pp. i–208, Dec. 1987, doi: 10.1002/j.2330-8516.1987.tb00244.x.

[17] U. Kulisch, *Computer Arithmetic and Validity: Theory, Implementation, and Applications*. Walter de Gruyter, 2013.

[18] J. Russ and K. E. Barlow, *British Bat Calls: A Guide to Species Identification*. Anchor Books, 2012.

[19] O. Mac Aodha *et al.*, “Bat detective—Deep learning tools for bat acoustic signal detection,” *PLOS Computational Biology*, vol. 14, no. 3, p. e1005995, Mar. 2018, doi: 10.1371/journal.pcbi.1005995.

[20] C. L. Walters, A. Collen, T. Lucas, K. Mroz, C. A. Sayer, and K. E. Jones, “Challenges of Using Bioacoustics to Globally Monitor Bats,” in *Bat Evolution, Ecology, and Conservation*, New York, NY: Springer New York, 2013, pp. 479–499. Available: http://dx.doi.org/10.1007/978-1-4614-7397-8_23

[21] K. E. Jones *et al.*, “Indicator Bats Program: A System for the Global Acoustic Monitoring of Bats,” in *Biodiversity Monitoring and Conservation*, Oxford, UK:

Wiley-Blackwell, 2013, pp. 211–247. Available: http://dx.doi.org/10.1002/9781118490747.ch10

[22] V. Stathopoulos, V. Zamora‐Gutierrez, K. E. Jones, and M. Girolami, "Bat echolocation call identification for biodiversity monitoring: a probabilistic approach," *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 67, no. 1, pp. 165–183, Feb. 2017, doi: 10.1111/rssc.12217.

[23] J. Rydell, S. Nyman, J. Eklöf, G. Jones, and D. Russo, "Testing the performances of automated identification of bat echolocation calls: A request for prudence," *Ecological Indicators*, vol. 78, pp. 416–420, Jul. 2017, doi: 10.1016/j.ecolind.2017.03.023.

[24] J. (Y) Stein and Stein, *Digital Signal Processing: A Computer Science Perspective*. Wiley-Interscience, 2000.

[25] N. Jayant, J. Johnston, and R. Safranek, "Signal compression based on models of human perception," *Proceedings of the IEEE*, vol. 81, no. 10, pp. 1385–1422, 1993, doi: 10.1109/5.241504.

[26] K. Sayood, *Introduction to Data Compression*. Elsevier Science, 2017.

[27] M. Mahoney, "Data Compression Explained," Apr. 15, 2013. http://mattmahoney.net/dc/dce.html

[28] B. Kim and Z. Rafii, "Lossy Audio Compression Identification," in *2018 26th European Signal Processing Conference (EUSIPCO)*, Sep. 2018, pp. 2459–2463.

[29] I. N. Herstein, *Topics in Algebra*, 2nd ed. John Wiley & Sons Incorporated, 1975.

[30] J. Watkinson, *The MPEG Handbook: MPEG-1, MPEG-2, MPEG-4*. Taylor & Francis, 2004.

[31] J. Coalson, "FLAC," *documentation*. https://xiph.org/flac/documentation_format_overview.html (accessed May 15, 2022).

[32] The Library of Congress, "MP3 (MPEG Layer III Audio Encoding)," May 03, 2017. https://www.loc.gov/preservation/digital/formats/fdd/fdd000012.shtml (accessed May 16, 2022).

[33] F. Ghido, "OptimFROG," *Main*. http://losslessaudio.org/ (accessed May 15, 2022).

[34] A. Djuric, “TTA Lossless Audio Codec - Format Description,” *Tau Software*. https://www.tausoft.org/en/true_audio_codec_format/ (accessed May 16, 2022).

[35] “WavPack Audio Compression.” https://www.wavpack.com/ (accessed May 16, 2022).

[36] “WavPack7z.” https://www.tc4shell.com/en/7zip/wavpack7z/ (accessed May 16, 2022).

[37] A. Diaz, “Xz format inadequate for long-term archiving.” http://lzip.nongnu.org/xz_inadequate.html (accessed May 16, 2022).

[38] “7z Format - 7-Zip Documentation.” https://documentation.help/7-Zip/7z.htm (accessed May 16, 2022).

[39] J. Cleary and I. Witten, “Data Compression Using Adaptive Coding and Partial String Matching,” *IEEE Transactions on Communications*, vol. 32, no. 4, pp. 396–402, 1984, doi: 10.1109/tcom.1984.1096090.

[40] PKWARE, Inc., .“ZIP File Format Specification,” Jul. 15, 2020. https://pkware.cachefly.net/webdocs/casestudies/APPNOTE.TXT (accessed May 16, 2022).

A. Appendix A

B.1. Questions

The purpose of this questionnaire is to assess the current state of ultrasonic audio data management, especially in regard to ultrasonic bat vocalizations. The goal is to create a novel data compression method for this purpose, as most audio data compression techniques are focussed on the human range of hearing and may not be suitable for ultrasonic audio as a result.

We define data compression as a process to reduce the size of a data file, often by either restructuring or modifying the data, for the purpose of making storage or transfer more efficient.

1. Briefly, what specifically is your job title/description, how long have you been doing it, and what is your current project/goal?

2. On average, how much bat/ultrasonic audio data do you generate for a project? How large are individual files on average/at most?

3. How impactful is the size/amount of audio data that you work with? Does storage or transfer of these files consume a problematic amount of storage space or time?

4. What software do you use to analyse this audio, and what functions do you primarily use it for? What can/can't be done automatically?

5. What percentage/how much of the average audio file is empty/useless/"noise"? How do you identify this noise? Is this noise removed somehow before/during analysis?

6. Does ultrasonic audio data have to be treated significantly differently from other audio data?

7. In what format do you generally store bat/ultrasonic audio data (e.g., full spectrum vs zero crossing, compressed vs uncompressed) and what file types do you commonly use (e.g., wav, zc, mp3, flac, zip)? Why?

8. How long do you generally store any given bat audio data and how often do you generally transfer significant amounts of it?

9. Would you value a way to reduce the size of the bat/ultrasonic audio data you work with?

10. Three important metrics for determining the value of a compression algorithm are:

· Size: How much smaller is the compressed file compared to the original?

· Speed: How much time does it take to compress a file?

· Accuracy: How much detail was lost during compression?

How do you value these metrics? Are there other metrics that are also important to you?

11. Do you know whether anybody has developed a data compression method focussed on ultrasonic audio before? If not, do you know why not?

12. It has been rumoured that zero crossing files may be phased out in the future. What have you heard about this, if anything?

Finally, would you be willing to provide us with sample files of bat/ultrasonic audio of the kind(s) you generally work with?

B.2. Interview A

Researcher A:
All right. I hope we've got that set up.

Interviewee:
Yep. That's fine.

Researcher A:
All right. So we are going to interview to you today about ultrasonic about vocalizations.

Interviewee:
mm-hmm <affirmative>

Researcher A:
We're hoping to develop a novel data compression method, and we wanted to get your opinion on the matter.

Interviewee:
mm-hmm <affirmative>

Researcher A:
So we've got some questions for you.

Interviewee:
Yep. Cool.

Researcher A:
Briefly, what is your job title and description? How long have you been doing it and what's your current project or goal?

Interviewee:
So I'm a PhD student. I've been doing this PhD since 2016 but I've been working with bats for 15, 20 years almost. And so now I'm my project. I, yeah, I use ultrasonic recording methods to monitor bats in different habitats populations. So primarily I'm using ultrasonic data and a lot of it.

Researcher A:
Yeah. We heard as much. You're with [website], right?

Interviewee:
Yes. So that's one of the projects. [Description of current project.] So, and we hope we'll be to continue that again, this year.

Researcher A:
Everyone I've shown that to thinks it's very interesting. So very nice. On average, how much ultrasound audio data do you usually generate for a project?

Interviewee:
Well, you can say that per detector per device per night, that it goes out if I'm recording for the entire night, it's 10 gigabytes.

Researcher A:
All right.

Interviewee:
And I am often deploying 20 of these per night, maybe over six, seven weeks. So,

Researcher A:
Oh, that's a large amount.

Interviewee:
Yeah, it's a large amount. So, I mean, I sit with currently with, I think I've got now my sixth hard drive, which is two terabytes and I also have to back up these things somewhere. So they're backed up either on our server here or backed up on the cloud. And of course it's really time consuming. Even to just to take them from one place to another is really time consuming. And I use a slightly faster computer for processing, so it runs on like 32 cores. So it's a little bit faster when it's processing the stuff for me, but of course it's really slow because it has to go through all the noise files as well. And this takes time. So yeah, anything to minimize this is really wanted, I think, by anyone who works with ultrasonic calls.

Researcher A:
I hope so. Yeah. And I think you basically asked, answered the next one. How impactful is the size? So pretty impactful and annoying.

Interviewee:
Yeah.

Researcher A:
What software do you use to analyze the audio? Like what do you, what do you use it for? What can you not do automatically?

Interviewee:
So at the minute I'm using Kaleidoscope, which is a purchased software from Wildlife Acoustics. I think I mentioned before that, I think they do have some form of compression option within the software. I've not used it, then I've not heard people talk

about it, but it could be worth checking out what they do. I don't know what it is there. Yeah. How they're handling it.

Researcher B:
I looked into it and like, they seem to have it, but it is, infuriatingly hard to find any details about it whatsoever. Like where it's used, how it works. We can find like two paragraphs explaining how it works, but nothing like where it's used or anything it's

Interviewee:
Yeah.

Researcher B:
Quite a mystery.

Interviewee:
Yeah. Other softwares I've used are... there's an open source, one called BatScope 4 and then there's BatSound, which is…

<redacted due to disclosure of personal information>

… BatSound. Which is one of these it's kind of like the industry standard. I guess people everywhere use it,

Researcher A:
Just jotting that down. Right. What percentage how much of the average audio file do you consider to be the noise and how do you identify it?

Interviewee:
Yeah, so there's a lot of noise cause I'm just recording continuously and the detectors that I use at the minute don't trigger for a bat. You can, so Wildlife Acoustics have these detectors, actually, you set certain thresholds. So you can say that I only want you to trigger if you really truly think it's a bat and only record what we think are bats, and then you get a lot less noise, but now people are moving to these detectors that I use called AudioMoths. They are super cheap, but they just record everything. That's why you get 10 gigabytes in a night. And so then there's a lot of noise. I would say, if you have say, I, I divide them into like 10, second sound files when I record. And maybe like, it could be like 10% or useful and the rest are noise.

Researcher A:
Yeah. That's more than I would think of. And I, so

Interviewee:
Yeah, because it, and it just depends on the site. If it's a really active site, maybe it's, it's slightly less, but there are no triggering settings in AudioMoth, which I've not actually

tested, but you can trigger a little bit, but you'll still get a lot of noise. But I should say with the noise, it's always, we always have to check those noise files, at least a certain percentage of them because sometimes there's, if it's a very quiet call, you might miss the, the software might miss the fact there's a bat there cuz Kaleidoscope will go through your files and say, this is bats and this is noise, but you still wanna check those noise files. I check 10% of them and then just delete them.

Researcher A:
All right. Just to filter out. That makes sense.

Interviewee:
Yeah.

Researcher A:
All right then. I guess that's not particularly relevant, but cuz you mostly deal with ultrasonic audio your data, right? Mm. Although I believe bats do a lot of chirping too. Just normal social calls and neuro ultrasonic. Are those particularly relevant to your work?

Interviewee:
Yeah. And some of those are, you know, audible of course. And, and, and also I should say, I mean we're often interested also in, in some of the audible signs because I worked in a project where we recorded both bats and birds. So we had somebody who we worked with, the bird data, someone who worked with the bat stuff. So you can record everything with these detectors. I mean, you get a lot of crickets. This is actually one thing that's really infuriating. If you have crickets they're really loud and they just dominate the sound files. So you get lots of these sound files.

Researcher A:
I never really hated crickets for it, but <laugh>,

Interviewee:
<Laugh> that, I mean, that could be interesting still for some people they might want to look at cricket cause you can identify the species of cricket by the pattern of the, the sonogram. So yeah.

Researcher A:
Do you yeah, so you say, would there be any difference in how you have to treat like your ultrasonic audio data, compared to if someone wanted to use a human hearing range audio data like the birds, do you treat the data any different?

Interviewee:
I guess at the minute it's I guess the softwares are more developed for auto-identification of ultrasonic signs for bats, particularly cuz now they have these

huge libraries and it's a bit, there's a lot more possibility to do this auto identification, I guess, with the, with birds and things at the minute, people are still just listening to the files actually and saying, it's that bird? It's this bird cuz it's not, they're not there yet. I don't think with bird auto identification. So I guess that's the difference is that I can do a lot of sort of semi-automatic processes. I can let a software do a lot of the hard work and I just do a bit of checking, with birds they are really checking everything,

Researcher A:
I guess it might be because you can listen to the birds in the first place.

Interviewee:
So yeah. Yeah. That helps <laugh>. Yeah.

Researcher A:
All right. What format do you usually store your ultrasonic audio data in?

Interviewee:
They are wav files

Researcher A:
Full spectrum then

Interviewee:
Mm-Hmm <affirmative> yeah full spectrum wav files.

Researcher A:
All right. Then that's quick one there. Why that? Just to make sure you don't lose any of the data?

Interviewee:
Because it's the format that these detectors record in. Oh, that's the primary reason. Yeah, and I mean, there's no reason for me to convert it because again, again, the softwares I guess are primarily handling wav files.

Researcher A:
Yeah,

Interviewee:
I'm not sure if they can handle other ones or not, I'm not sure

Researcher A:
I heard they sometimes use zero crossing where they it's basically very rare. Yeah.

Interviewee:

Yeah. That is for Anabat, which is particular company. They produce these yeah. Zero. You, you can also do it with some other change but Anabat's the primary product that you do that with and you lose, you get a lot, you sort of lose information with this zero crossing. So it means that the, the, yeah, the files are kind of compressed in a sense and easier to handle, but then it's quite difficult to identify some species. So there's a particular group called the Myotis group. And I think with zero crossing, it can be quite difficult to separate them. It's already really difficult anyway. But then with zero crossing, it becomes worse.

Researcher A:
Yeah. Currently only they only record like one frequency with the zero crossing don't they? So that's,

Researcher B:
You can't even tell like how many bats are around just with zero crossing files. Just how many times a noise has been made.

Interviewee:
Yeah. But it's the same with, I mean, what I do, I can't say that there were 50 bats because there's no way to say how many individuals, so what you tend to say is that in a ten second period, if you have more than two pulses of a call, then you can say a bat pass happened, as in a bat flew past your microphone. And then you say how many bat passes there were. And that does is a rough proxy for how many bats, but you can never say how many bats.

Researcher A:
They might just be doing circles around you.

Interviewee:
Exactly. They can do this, just pass your microphone constantly.

Researcher A:
Once they learn where the microphone is, they're just messing with you.

Interviewee:
<Laugh> yeah. So yeah, there are ways to try and people have tried to come up with ways to get numbers, but it's very difficult.

Researcher A:
Yeah.

Interviewee:
Modeling.

Researcher B:
So bats of the same species roughly are very hard to tell apart just from their sounds?

Interviewee:
Yeah. You couldn't say individuals. No, mm-hmm <affirmative>

Researcher B:
Okay.

Researcher A:
All right. How long do you usually store your like a given recording and how often do you usually like have to transfer it around?

Interviewee:
I mean, they're stored forever because I mean it's publicly funded data. And I would say moving it, I have to take it yeah. From initially from the device, which is stored in an SD card. And then it has to go onto a hard drive probably first of all in the field. And then I have to back it up somewhere more secure, like a cloud solution. I, I use Box, which is very clunky and very slow. But that's what we're supposed to use at university. We're not really allowed to use data storage, like cloud solutions that are outside of Sweden. So no American based ones. Box apparently in theory is, I don't know, you have weird policies, but anyway, we use Box and I also have a server also it's very slow. And then I guess I have to move it. When I'm using my processing computer, it's not my computer, it's one for the whole department. So then I have to transfer it onto there normally by hard drive. And then I gotta transfer back the outputs that I get as well, A little bit of transferring

Researcher A:
On that one. Yeah. <Affirmative> several times with several terabytes. Seems a lot. Yeah. All right. I guess this is our biggest one for us. So normally when you talk about compressing something, it, it, you get to speed, size, and accuracy as the main criteria. So how fast can you compress it, how small does it get, and how much do you lose when you do that? How would you like rank those importance? Is it okay to lose some accuracy in order to make it really quite small?

Interviewee:
I mean, I guess it depends how much you lose. But, that's the tricky thing. I mean, for me, I think some speed would be good. Absolutely. but as I said with some species, it's like, you don't mind so much cuz you know, if you use a little bit of information, you'll still get, you'll be able to get to the species identification, but for some of them you really do need good quality data. So I guess, I guess I'd have to almost see some examples of how that might look to kind of really judge, because

Researcher A:

Have you, have you heard of MP3?

Interviewee:
Yes.

Researcher A:
These are usually considered to be lossy, which means that they've thrown out a lot of the accuracy and a lot of people can't really tell compared to like a .wav file is like, so that would be the kind of thing we're thinking of. Cause you can get an MP3 down to 5% of the original size and people won't really notice. Mm. But it can also make it more difficult for like a computer analysis thing. Cause you're throwing out things that people are listening to the computer can still tell something's a little off. Would that be a, would that level of like compression be alright? Where it's <garbled> to listen to.

Interviewee:
Yeah. I mean, cause I'm doing two things with this data, I guess it's one that I, I do listen, but I would say I use my ear a little bit, but I actually, I'm actually more looking visually at how these frequencies who distribute it in time and across the frequency range. So you're looking at the visual representation of it. And I'm also looking at what the, the, the metrics the software have kicked out. So what was the, what was the highest frequency? What was the lowest frequency? Where was the energy with the, the most where the energy was the highest? What was that frequency? These are certain parameters that are really important when you're looking at bats species identification. So as long as you're not losing the top frequencies, the bottom frequencies, these most powerful frequencies, I think it's okay. But if you start to lose something on the top and bottom, it could really affect identification for some species.

Researcher A:
All right. I think we might want to press you a little bit to get like, like some kind of complete list for what's important there. If you have just slow. Yeah. I can something to point me to,

Interviewee:
I can maybe just take I'm trying to think. I could probably take a there's like maybe I can get an image out of one of the books that I use of, of these key parameters that we look at.

Researcher A:
That would be wonderful.

Interviewee:
But it is really like looking at these kind of max min values. Also like the start time and the end time, how long is the call in total? It's the sort of information that we're looking for? I guess what, what might be a problem is that if you have these very weak calls,

maybe compression might make it really hard for the computer to extract those values. Then you've gotta think about how important are these really weak calls anyway which is something we always try and think about. It's that if it's really weak was the bat really where you think it was, could it have been quite far away and actually it's not really useful.

Researcher A:
So I see.

Interviewee:
Yeah. So it's, it's, it's tricky. <Laugh> but I can send you a list of sort of the key things we're looking at. Yeah.

Researcher A:
I'd appreciate that. Cuz if we're gonna make our own compression, we can make sure we keep the key stuff in.

Interviewee:
Yeah.

Researcher A:
And then toward this, you would say like, since they're stored forever, I assume that you would quite like to get quite small that would be quite important. And then if it takes a little while to get to that size, but then the storage is easier, that's probably be maybe not directly for your work, but just as a general uses of resources would probably be the angle?

Interviewee:
Yeah. Yeah, I think so.

Researcher A:
All right. I guess if that's the most important Ben, anything I missed on that one?

Researcher B:
No, I suppose it's up to us to figure out the exact numbers of all that, but I suppose just kind of like rough rule of thumb if it, if you could like spend a week to reduce the file size by half and like all of the, you know, sending it around and transmitting the data is like now ha like twice as fast, would that be worth it or is a week just like sending it around twice as fast? Just like less than a week. Yeah, it's kind of,

Interviewee:
Yeah. I mean it could be worth it because I mean I already maybe spent maybe like for the processing of like a whole survey, like where a whole summer's worth of data might take me, like the, the first processing might take a couple of weeks. So actually in the

scheme of things, it's a huge amount of time, extra as long as the quality is still there. Yeah.

Researcher B:
Mm-Hmm <affirmative> okay. Yeah. That's a good rough, number to start with. So thank you very much.

Researcher A:
All right. And then I guess some rumors, I suppose, do you know if anyone else is like developing any other data compression and methods? I think you mentioned the wildlife acoustic. Yeah. I think it's a dot wac. Yeah. You the anything else? That's

Interviewee:
The only one I'm aware of and I've not really heard it been talked about so much more, the people talk about like, oh, I've got so much data is kind of the complaint, but not that there's any real solutions out there.

Researcher B:
Yeah. That's a lot of what we're finding too. Just a lot of like, someone should do something about this, dot dot dot

Interviewee:
The only other companies, I guess that could be looking at it. It could be worth. Cause I mean, they're kind of on top of it as a French or, well, they're not a company they're they're they are a research group, actually. SonoChiro. Have you heard of SonoChiro?

Researcher A:
I don't believe so, no.

Interviewee:
No? They're a French group and they have been doing lots of stuff. And they have their own software. And I could imagine if anyone was gonna be looking at it, it would be them. Cuz they seem to be quite on top of everything when it comes to like <garbled>

Researcher A:
Would their, would their solution end up being proprietary?

Interviewee:
You have to. I mean it's a, they, they software is they sell it. Yeah.

Researcher A:
Alright then. So I guess it's part of the university. I don't think we would be allowed to use that then.

Interviewee:
They are. I think they are a university. Yeah. I dunno whether they also have a small company cuz they, cause they are actually selling this product. They, they may well have a small company and they make a detector called well, I think they at least work with the company who make the detector Dodotronic. And it's also quite a low cost detector Dodotronic.

Researcher A:
I think I found it, a microphone

Interviewee:
They're actually doing quite a lot there, but I, I, Wildlife Acoustics are kind of the main big company. Petterson. I mean, they're also big, but they're so traditional that I can't imagine they're working with that. They're very much, they're a bit more old school maybe than that's like Wildlife Acoustics. And then you also have the people around AudioMoth people who made AudioMoth. Now they're also a research group. They're not really so much for-profit, but they may well be looking at this kind of thing too, but I have not heard about it.

Researcher A:
All right. And then, Hey, this is actually very rumor. I've heard a rumor that zero crossing files are not, are gonna be phased out in the future. Any idea about that?

Interviewee:
I don't know. I've not heard. I mean, I, I was at a workshop some years ago with Anabat were there and they very much promote zero crossing. So I mean that's a huge part of their business. So I, I don't know, I've not heard that.

Researcher A:
Heard from someone else I've spoken to about this and it was very off the cuff, so

Interviewee:
Yeah. But I know people are moving more and more to like full spectrum recording people want, that

Researcher A:
Makes sense to me.

Interviewee:
Yeah.

Researcher A:
All right. I think that's all the ones I have written down. Anything I missed?

Researcher B:
Nothing in particular, I suppose. Just kind of, because we're not as familiar with this field as you are obviously. Is there anything else you would want like that we may be overlooking from audio compression and just like, make sure you do this or don't do this, that just kind of like an outsider wouldn't really be able to just kind of guess from looking in?

Interviewee:
I can't think of anything to, I mean, me, I think the main thing is people will want the quality to be there still. Yeah. Because it is it's difficult anyway to work with these these files and to get, to get enough good data from them. But no, I can't think of anything if I think of anything, I could always ping you an email, but yeah.

Researcher B:
Nothing

Interviewee:
Comes to my name.

Researcher B:
Yeah.

<redacted due to disclosure of personal information>

Researcher A:
Yeah. this has been very good. Great. Thank you very much for your time.

Interviewee:
Yeah. It's really interesting. It's good that somebody's looking at this. I think cuz it's really needed

Researcher B:
When, when we started looking into this, we were kind of nervous that like, oh this like is such a common problem. Of course there's gonna be like dozens and dozens of like way better people than us trying to solve it. And just like what we find it really seems just to be like, nobody's really tried yet. So we're hopeful we can actually add something pretty helpful to this.

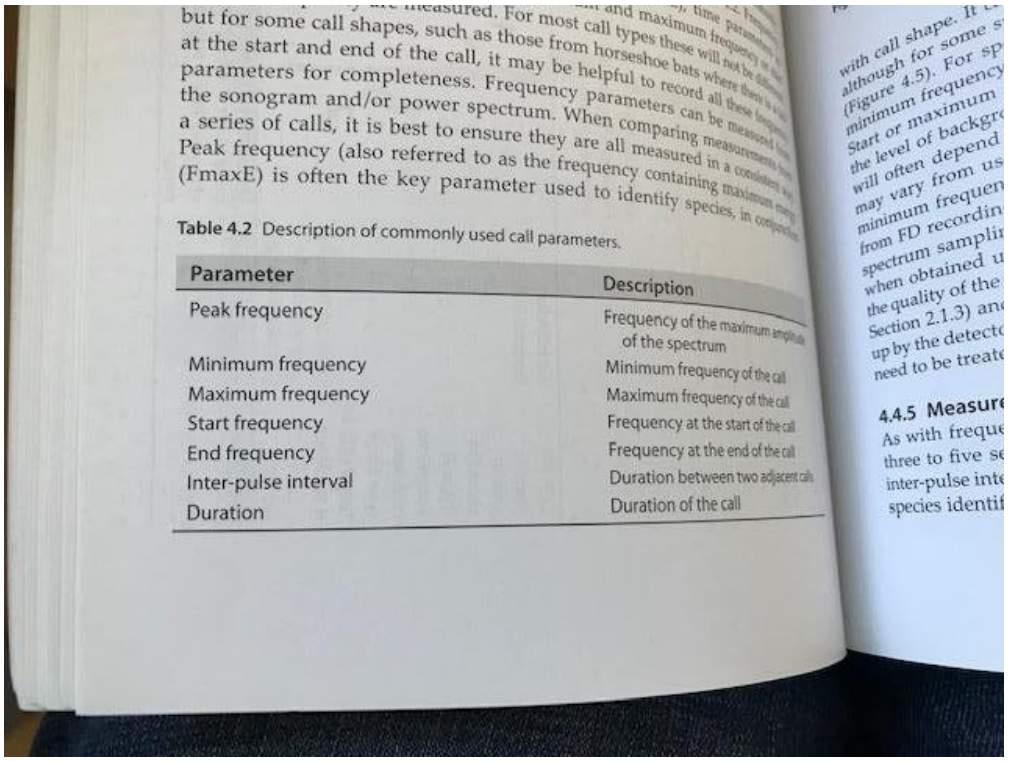
Interviewee:
Yeah. Excellent.

Researcher B:
Thank you very much.

Interviewee:
Yeah. Thank you. Good luck with everything.

Researcher A:
You too. Bye.

Interviewee:
Okay. Bye.



but for some call shapes, such as those from horseshoe bats where there is a
at the start and end of the call, it may be helpful to record all these
parameters for completeness. Frequency parameters can be measured
the sonogram and/or power spectrum. When comparing measurements
a series of calls, it is best to ensure they are all measured in a consistent
Peak frequency (also referred to as the frequency containing maximum
(FmaxE) is often the key parameter used to identify species, in conjunction

Table 4.2 Description of commonly used call parameters.

Parameter	Description
Peak frequency	Frequency of the maximum amplitude of the spectrum
Minimum frequency	Minimum frequency of the call
Maximum frequency	Maximum frequency of the call
Start frequency	Frequency at the start of the call
End frequency	Frequency at the end of the call
Inter-pulse interval	Duration between two adjacent calls
Duration	Duration of the call

Figure A.1:Picture of Figure 4.2 from British Bat Calls: A Guide to Species Identification [18], provided by Interviewee A after the interview.

B.3. Interview B

The purpose of this questionnaire is to assess the current state of ultrasonic audio data management, especially in regard to ultrasonic bat vocalizations. The goal is to create a novel data compression method for this purpose, as most audio data compression techniques are focused on the human range of hearing and may not be suitable for ultrasonic audio as a result. We define data compression as a process to reduce the size of a data file, often by either
restructuring or modifying the data, for the purpose of making storage or transfer more efficient.

1. Briefly, what specifically is your job title/description, how long have you been doing it, and what is your current project/goal?

I'm a bat biologist (since [year]), currently doing postdoctoral research at [university]. My two main focuses are animal behavior and conservation. Both aspects of my work make extensive use of bioacoustics. One of my current projects is an investigation on individuality of bats' use of echolocation and its importance to foraging strategies.

2. On average, how much bat/ultrasonic audio data do you generate for a project? How large are individual files on average/at most?

A typical project can generate anywhere from 1 to 6 terabytes, long projects can add up to more than that. I make sure the individual files are not too large so that analysis is easier – I limit recording length to 4 seconds, which at a sampling rate of 375kHz creates files that are about 3 megabytes in size.
I work generally with .wav files without compression as all the information is important. For some projects I use zero-crossing files.

3. How impactful is the size/amount of audio data that you work with? Does storage or transfer of these files consume a problematic amount of storage space or time?

Impacts of size/amount for work: a. much of my work is done on the spectrogram rather than the oscillogram, and the larger the file the longer it takes to generate a spectrogram; b. much of the analysis that can be automated has to be manually audited, again – time-consuming due to both file size and amount of files;
Impacts of size/amount for storage or transfer: Storage is becoming less of a problem though still relatively expensive (US$300-400 for a 4TB hard-drive). Transfer is where the real issue lies, and at the moment the best way to transfer large amounts of audio files is to load them on a hard-drive and physically mail them. This of course is also a time-wasting aspect.

4. What software do you use to analyse this audio, and what functions do you primarily use it for? What can/can't be done automatically?

For .wav files I use SasLab Pro by Avisoft Bioacoustics. I'm often interested in call duration, intervals, amplitude, and frequency distributions. Depending on the quality of the recording (which itself depends on project conditions, i.e. field/lab/controlled etc.), these can be automated to a degree.
For field surveys that need species ID I use for Sonobat for .wav files and Analook for ZC files. Both can run automatically (depending on region and the amount of acoustic overlap between local species) to a certain degree of accuracy and both provide much better results after post processing manual auditing.

5. What percentage/how much of the average audio file is empty/useless/"noise"? How do you identify this noise? Is this noise removed somehow before/during analysis?

First, noise or empty is not useless as it provides information about intervals between calls which is very important. “Noise” can be different things: it can be biological in origin or anthropogenic, and these may also be of use. In most cases, when you set the recording regime there are three strategies to reduce unwanted noise: 1. Recording schedule: setting the recorder to sleep during the day excludes a lot of anthropogenic and biological non-bat noise. 2. High-pass filter: most bat calls are in the ultrasonic range so setting an appropriate HP filter (adapted to local species frequencies) excludes most non-bat biological sounds (crickets/katydids etc. do get recorded). 3. Gain/amplitude threshold: gets rid of most wind-generated sounds or low-quality recordings.
The same strategies can also be applied during analysis to exclude some of the noise that escaped recording regime filters.
Most analysis software and some recorders have the ability to cut silence out of the files while preserving the time-stamp of the calls and in this way, they preserve information about intervals. Personally, I do not use these features because I like to have an accurate visualization, but this is only me and this feature can be very useful to reduce file sizes. The feature works according to a pre-determined amplitude threshold.

6. Does ultrasonic audio data have to be treated significantly differently from other audio data?

No, but in order to record ultrasound accurately you must have a very high sampling rate (most people use 250—375kHz) which significantly increases file size.

7. In what format do you generally store bat/ultrasonic audio data (e.g., full spectrum vs zero crossing, compressed vs uncompressed) and what file types do you commonly use (e.g., wav, zc, mp3, flac, zip)? Why?

Mostly I use and store full-spectrum, uncompressed .wav files because they conserve a accurately as possible all the acoustic features of the calls.
I use ZC as well, mostly because I still use older recorders that ZC is their only output. They are actually not bad for long-term acoustic field surveys because storage is not an issue and because it is relatively easy to create filters for species ID without needing programming skills. That being said, I do not convert full-spectrum recording to ZC.

8. How long do you generally store any given bat audio data and how often do you generally transfer significant amounts of it?

Store – indefinitely, depending on project, after getting rid of non-relevant files.
Transfer – that really depends on the project. Long term survey results are usually transferred between several recipients once or twice a year, but every

group/researcher/practitioner usually develop their own working strategy, so I can't say this is representative of anything.

9. Would you value a way to reduce the size of the bat/ultrasonic audio data you work with?

Obviously, but not at the expense of introducing distortions of any kind.

10. Three important metrics for determining the value of a compression algorithm are:
• Size: How much smaller is the compressed file compared to the original?
• Speed: How much time does it take to compress a file?
• Accuracy: How much detail was lost during compression?
How do you value these metrics? Are there other metrics that are also important to you?

Accuracy is of utmost importance and completely trumps the other two in my opinion. After that, size would be more important than time for me.

11. Do you know whether anybody has developed a data compression method focused on ultrasonic audio before? If not, do you know why not?

I do not know.

12. It has been rumoured that zero crossing files may be phased out in the future. What have you heard about this, if anything?

This definitely seems to be the direction, for a number of reasons: 1. Memory cards constantly get cheaper and with higher storage capacities. 2. Full spectrum recorders are no longer more expensive than ZC ones, and real-time recording is no longer an issue due to improved buffering capabilities. 3. Automated species ID tools for full-spectrum recordings are improving.

Finally, would you be willing to provide us with sample files of bat/ultrasonic audio of the kind(s) you generally work with?

Attached to the email are several files.

B.4. Interview C
Data compression consultation

Contributors: <redacted> (Director), <redacted> (Principal Ecologist), <redacted> (Senior Ecologist), <redacted> (Consultant)

Responding to the following brief and associated questions:

We define data compression as a process to reduce the size of a data file, often by either restructuring or modifying the data, for the purpose of making storage or transfer more efficient.

Questions	Combined responses
1. Briefly, what specifically is your job title/description, how long have you been doing it, and what is your current project/goal?	The contributors are professional ecological consultants of between 5 and 25+ years, who have worked on projects requiring large-scale data sets (wind farm, road scheme, new nuclear). We have several ongoing projects.
2. On average, how much bat/ultrasonic audio data do you generate for a project? How large are individual files on average/at most?	Amount of data very variable. · New nuclear (oldest project generating data set, dating back to 2013/14): 7m recordings, 2TB (collected on the older SM2s). · [Location] road scheme (later): similar (but SM4s) · [Location] road scheme: 3TB · Windfarm (2021): 4TB Individual files vary from 1.5 to 3MB.
3. How impactful is the size/amount of audio data that you work with? Does storage or transfer of these files consume a problematic amount of storage space or time?	Yes, storage and transfer are both problematic! However, it is worth noting that the presumption is that a RAW Dataset will be retained in every circumstance so that the files as originally collected are available to a third party in case of dispute or change.
4. What software do you use to analyse this audio, and what functions do you primarily use it for? What can/can't be done automatically?	Batch processing: BTO Acoustic Pipeline or SonoBat Viewing/manual analysis: SonoBat, Kaleidoscope – or BatExplorer/Analook when using detectors that were developed with that software. Have used BatSound as a viewer in the past.

5. What percentage/how much of the average audio file is empty/useless/"noise"? How do you identify this noise? Is this noise removed somehow before/during analysis?	Several points here: (i) files that are just 'noise' and (ii) the 'noise only' parts of recordings that also contain bat calls... (i) Percentage of files that are 'just noise' can be minimised by making sure detectors are appropriately located, but there are environmental factors that vary from site to site that can't be controlled through location (including weather). Those elements are best reduced by setting appropriate filters on recording. Such files can be identified and labelled as such by the software used for sound analysis and even sorted out into separate folder(s), but they are retained – see response to Q3. (ii) The 'noise only' parts of recordings that also contain bat calls tend to be short (and at the end of each recording), and are reduced by appropriate trigger settings.
6. Does ultrasonic audio data have to be treated significantly differently from other audio data?	While our equipment may collect non-ultrasonic data, we don't do anything with that data, nor set out to collect it deliberately.
7. In what format do you generally store bat/ultrasonic audio data (e.g., full spectrum vs zero crossing, compressed vs uncompressed) and what file types do you commonly use (e.g., wav, zc, mp3, flac, zip)? Why?	All data is full spectrum uncompressed these days and stored as .wav files retained – see response to Q3. For the oldest project we were involved with, we collected files that were compressed (.wac) to save storage space, because in those days detectors only had room for two 32GB cards. We then had to decompress them and store as both raw (compressed) and decompressed (pre-analysis) versions. Now that detectors can support larger SD cards, there is no need to collect as .wac and decompress. Whilst the decompression process could run in the background, it still required some interaction, which was relatively quick at each point but intermittent and disruptive.
8. How long do you generally store any given bat audio data and how often do you generally transfer significant amounts of it?	This is a commercial/legal decision; tends to be seven years post-completion (not post-collection). Once analysed, it can sit in the background somewhere; doesn't need to be instantly accessible. However, the presumption to keep raw (unaltered)

	data may limit the value of any compression software.
9. Would you value a way to reduce the size of the bat/ultrasonic audio data you work with?	Yes, but need to deal with point above (i.e. extensive testing to ensure that the product stored is as good as the raw data would be, to justify not keeping the raw data itself), and data needs to be retrievable in the .wav format that all analysis software uses.
10. Three important metrics for determining the value of a compression algorithm are: · Size: How much smaller is the compressed file compared to the original? · Speed: How much time does it take to compress a file? · Accuracy: How much detail was lost during compression? How do you value these metrics? Are there other metrics that are also important to you?	Accuracy and the raw storage issue (as noted in the previous answer). Important that the compression process does not lose any components of the bat calls or associated metadata.
11. Do you know whether anybody has developed a data compression method focused on ultrasonic audio before? If not, do you know why not?	As noted, Kaleidoscope allows the collection of data in a compressed format (.wac), but the files then need to be decompressed for analysis (which is not what you are suggesting here). We don't know of any other attempts; perhaps the answer to Q9 is the main reason why!
12. It has been rumoured that zero crossing files may be phased out in the future. What have you heard about this, if anything?	We don't use ZC files as we believe this format loses some elements of the call and therefore decreases the ability of analysts (machine or human) to provide an accurate identification.

We would be willing to provide you with sample files of bat/ultrasonic audio.

A.4.1 Interview C Follow-Up Questions

Presumption re keeping raw/unaltered files: good question.

Who presumes/demands this and what are the terms of it?	Most companies will have a retention policy for data. In addition, the revised bat survey guidelines (due later in 2022) will say: *The value of recording bat activity is that there is an auditable record of work carried out. Bat echolocation data collected during bat surveys should be stored in case this auditable record requires later scrutiny. The data should be stored in its raw form and in its entirety."* (but I should say that I added the last sentence to the draft, and that hasn't yet been approved so there is time to modify/caveat it!).
It is for legal reasons or company policy (or both)?	The reason is to enable a third-party to take the files and reanalyse them, potentially using different software. This has been necessary from time to time where analysis is challenged, particularly on contentious schemes.
Is it defined somewhere how precisely stored files need to match the originals, or is it more a matter of "proving" that the original and the compressed/decompressed versions hold equal value?	As noted above, the files would need to be available for reanalysis, potentially using different software (I have been working on a project which recently had to request a third-party put manually analysed data through proprietary 'auto-ID' software because of lack of confidence in the original identifications). This might not have ben possible if the original files had not been retained. There is a risk, when 'mucking about' with files, that they will become corrupted or otherwise changed, rendering reanalysis impossible. If evidence can be supplied that the original and the compressed/decompressed versions hold equal value, then maybe that would be OK. However, software can rapidly change or becomes unavailable. I think .wav files will be with us for a while, but anything that has a small market share may not, so people with responsibility for storing data may take a little convincing! As an example, we used SonoChiro software to auto-ID bat sound files back in 2013/14, but the SonoChiro designer moved on, and there are no functioning versions of SonoChiro left...

B. Appendix B

B.1. Compression Powershell Script

```
for (($i = 1); $i -lt 11; $i++) {
  Compress-Dataset -setNumber 1 -iterationNumber $i
  Compress-Dataset -setNumber 2 -iterationNumber $i
  Compress-Dataset -setNumber 3 -iterationNumber $i
  Compress-Dataset -setNumber 4 -iterationNumber $i
}
function Compress-Dataset {
  param ($setNumber, $iterationNumber)
  Set-Location $PSScriptRoot
  Remove-Item Output\*

  $dataSetPath = "TestSet" + $setNumber
  $files = Get-ChildItem -Path . -Filter $dataSetPath\*.wav -File

  # Create the Excel file that records the measured times if it doesn't
exist
   $outputFile = "output" + $setNumber + "(" + $iterationNumber +
").csv"
  if (!(Test-Path $outputFile)) {
    New-Item -path . -name $outputFile -type "file"
  }

  # Print uncompressed file size
   $inputFileSize = ($files | Measure-Object -Property Length -Sum |
Select-Object -expand sum)
  "Original WAV size (bytes)`t" + $inputFileSize | Out-File $outputFile

  #Table Headers
   "Type`tTime`tSize (bytes)`tCompressed Ratio" | Out-File $outputFile
-Append

  # FLAC
  $time = Measure-Command {
    $files | ForEach-Object {
```

```
                & .\Exes\ffmpeg.exe -hide_banner -loglevel error -i
$dataSetPath\$_ Output\$_.flac
      }
    }
       $fileSize = (Get-ChildItem -Filter Output\*.flac -File |
Measure-Object -Property Length -Sum | Select-Object -expand sum)
    "FLAC`t" + $time.ToString() + "`t" + $fileSize + "`t" + ($fileSize /
$inputFileSize) | Out-File $outputFile -Append
    Write-Host "FLAC Finished"

    # MonkeyAudio
    $time = Measure-Command {
      $files | ForEach-Object {
        & Exes\MAC.exe .\$dataSetPath\$_ Output\$_.ape -c2000
      }
    }
       $fileSize = (Get-ChildItem -Filter Output\*.ape -File |
Measure-Object -Property Length -Sum | Select-Object -expand sum)
     "MonkeyAudio`t" + $time.ToString() + "`t" + $fileSize + "`t" +
($fileSize / $inputFileSize) | Out-File $outputFile -Append
    Write-Host "Monkey's Audio Finished"

    # MP3
    $time = Measure-Command {
      $files | ForEach-Object {
                & Exes\ffmpeg.exe -hide_banner -loglevel error -i
$dataSetPath\$_ Output\$_.mp3
      }
    }
       $fileSize = (Get-ChildItem -Filter Output\*.mp3 -File |
Measure-Object -Property Length -Sum | Select-Object -expand sum)
    "MP3`t" + $time.ToString() + "`t" + $fileSize + "`t" + ($fileSize /
$inputFileSize) | Out-File $outputFile -Append
    Write-Host "MP3 Finished"

    # OptimFROG OFR
    $time = Measure-Command {
```

```
      $files | ForEach-Object {
            &  Exes\ofr.exe  --silent  --encode  $dataSetPath\$_  --output
Output\$_.ofr
      }
    }
       $fileSize   =   (Get-ChildItem   -Filter   Output\*.ofr   -File   |
Measure-Object -Property Length -Sum | Select-Object  -expand sum)
     "OptimFROG  OFR`t"  +  $time.ToString()  +  "`t"  +  $fileSize  +  "`t"  +
($fileSize / $inputFileSize) | Out-File $outputFile -Append
    Write-Host "OptimFROG OFR Finished"

    # TTA
    $time = Measure-Command {
      $files | ForEach-Object {
        & Exes\tta.exe -e $dataSetPath\$_ Output\$_.tta
      }
    }
       $fileSize   =   (Get-ChildItem   -Filter   Output\*.tta   -File   |
Measure-Object -Property Length -Sum | Select-Object  -expand sum)
     "TTA`t" + $time.ToString() + "`t" + $fileSize + "`t" + ($fileSize /
$inputFileSize) | Out-File $outputFile -Append
    Write-Host "TTA Finished"

    # WavPack
    $time = Measure-Command {
      $files | ForEach-Object {
        & Exes\wavpack.exe -q $dataSetPath\$_ Output\$_.wv
      }
    }
    $fileSize = (Get-ChildItem -Filter Output\*.wv -File | Measure-Object
-Property Length -Sum | Select-Object  -expand sum)
    "WavPack`t" + $time.ToString() + "`t" + $fileSize + "`t" + ($fileSize
/ $inputFileSize) | Out-File $outputFile -Append
    Write-Host "WavPack Finished"

    Set-Location $PSScriptRoot\$dataSetPath
    # 7Zip (LZMA2) NonSolid
```

```
    $time = Measure-Command {
        &  ..\Exes\7z.exe  a  -t7z  ..\Output\LZMANonSolid.7z  $files  -bb0
-ms=off
    }
     $fileSize  =  (Get-Item  ..\Output\LZMANonSolid.7z  |  Measure-Object
-Property Length -Sum | Select-Object -expand sum)
    "7-Zip (LZMA2)  NonSolid`t" + $time.ToString() + "`t" + $fileSize +
"`t" + ($fileSize / $inputFileSize) | Out-File ..\$outputFile -Append
    Write-Host "7Zip (LZMA2) NonSolid Finished"

    # 7Zip (LZMA2) Solid
    $time = Measure-Command {
      & ..\Exes\7z.exe a -t7z ..\Output\LZMASolid.7z $files -bb0 -ms=on
    }
      $fileSize  =  (Get-Item  ..\Output\LZMASolid.7z  |  Measure-Object
-Property Length -Sum | Select-Object -expand sum)
    "7-Zip (LZMA2)  Solid`t" + $time.ToString() + "`t" + $fileSize + "`t"
+ ($fileSize / $inputFileSize) | Out-File ..\$outputFile -Append
    Write-Host "7-Zip (LZMA2) Solid Finished"

    # 7Zip (PPMd) Solid
    $time = Measure-Command {
       & ..\Exes\7z.exe a -t7z ..\Output\PPMdSolid.7z $files -bb0 -ms=on
-mm=PPMd
    }
      $fileSize  =  (Get-Item  ..\Output\PPMdSolid.7z  |  Measure-Object
-Property Length -Sum | Select-Object -expand sum)
    "7-Zip (PPMd) Solid`t" + $time.ToString() + "`t" + $fileSize + "`t" +
($fileSize / $inputFileSize) | Out-File ..\$outputFile -Append
    Write-Host "7-Zip (PPMd) Solid Finished"

    # WavPack7z
    $time = Measure-Command {
        &  ..\Exes\7z.exe  a  -t7z  ..\Output\WavPack7zSolid.7z  $files  -bb0
-ms=on -myx=0 -m0=WavPack2 -m1=lzma:d20 -mb0s1:1
    }
```

```
    $fileSize = (Get-Item ..\Output\WavPack7zSolid.7z | Measure-Object
-Property Length -Sum | Select-Object -expand sum)
    "WavPack7z Solid`t" + $time.ToString() + "`t" + $fileSize + "`t" +
($fileSize / $inputFileSize) | Out-File ..\$outputFile -Append
    Write-Host "WavPack7z Solid Finished"

    # ZIP
    $time = Measure-Command {
      & Compress-Archive $files ..\Output\Zip.zip
    }
    $fileSize = (Get-Item ..\Output\Zip.zip | Measure-Object -Property
Length -Sum | Select-Object -expand sum)
    "ZIP`t" + $time.ToString() + "`t" + $fileSize + "`t" + ($fileSize /
$inputFileSize) | Out-File ..\$outputFile -Append
    Write-Host "ZIP Finished"
  }
```

B.2. Sample Analysis Python Script

```
import sys
import glob
import numpy
import librosa

ratioList = []
for file_path in glob.glob("*.wav"):
    samples, sampling_rate = librosa.load(file_path, sr = None)
    rangeStDevRatio = (max(samples) - min(samples)) / numpy.std(samples)
    print(file_path, " Range/STDEV: ", rangeStDevRatio)
    ratioList.append(rangeStDevRatio)
print("Average of all: ", numpy.mean(ratioList))
```