IMPACT OF LOWER QUALITY AUDIO

Project Echo

Abstract

After investigating the effect of Frequency Band Removal Method on Model Accuracy, in this report, I will move on to investigate the Impact of Lower Audio Quality on training. The data that I have used in the previous experiment will also be used to test in this digestion.

Nhat Minh Dang

222172836

Contents

A) Introduction:

Project Echo was founded in Trimester 3 of 2022 by Stephan Kokkas, Andrew Kudilczak and Daniel Gladman. The project aims to support global conservationists by developing advanced audio classification systems that can facilitate the non-intrusive monitoring, discovery and tracking endangered species and their predators within their natural habitats. The system of IoT will collect data and transmit to a center server via an API, where an AI-driven model classifies species and record vital data. Then, the Human Machine Interface (provides real time visualization of simulated animals and vocalization events on an interactive map.

As the project develops, more and more audio data are collected, increasing the current capacity. These audio files are varied in frequency, quality, duration, etc. In this task, I will digest deeper into utilizing the size of audio files using multiple Audio Quality Lowering Methods

B) Definitions and Concepts:

1) Elements of Audio Size:

The size of audio plays a vital role in resource consumption. The larger the size of data, the larger the use of resources to transfer and process data. There are several factors that determines the size of audio:

- Sampling rate: The number of audio samples captured per second, measured in Hertz (Hz). Higher sampling rates captures more detail from audio signals and increases the file size.
- Bit Depth: The number of bits used to represent each audio samples. Higher bit depth increases dynamic range and audio fidelity but also increases the file size.
- Number of Channels: The number of audio channels in the recording (mono, stereo, surround sound, etc). Stereo audio is approximately twice the size of mono audio while surround sound further increases file size.
- Audio Duration: The length of audio file in seconds or minutes. Longer audio files contain more data and result in larger file sizes.
- Audio Format: The encoding and compression method used to store the audio data (wav, mp3, aac, ect). While uncompressed formats (wav, aiff) are larger as they store more raw data, lossy compressed format (mp3, aac) has much smaller size by discarding some audio data with trade-off in quality.
- Compression bitrate: The amount of audio data processed per second, typically measured in kilobits per second (kbps).
- Metadata: Additional information embedded in the audio file such as artist name, album details and cover art.

Formula for Uncompressed Audio File Size:

File size (bytes) = Sampling Rate x Bit Depth x Number of Channels x Duration (seconds) : 8.

2) Audio Quality Lower Methods:

a) Reduce Bit Depth:

Reduce bit depth method is used to reduce the precision of the audio signal by limiting the number of distinct amplitude levels that can be represented. This can help to reduce the file size or simulate lower-quality audio for specific applications, such as retro-style audio processing or reducing computational load. In the provided code, the system takes an audio file as input and a desired bit depth (e.g., 8 or 4 bits). It first loads the audio using the librosa library. Then, it calculates the maximum value that can be represented with the specified bit depth (2^(bit_depth - 1) - 1) and scales the audio signal accordingly. The audio is then quantized by rounding the values to the nearest integer corresponding to the new bit depth, followed by normalization to ensure that the values remain within the valid audio range of -1.0 to 1.0. Finally, the function returns the reduced bit-depth audio signal along with the sampling rate. This method allows you to simulate lower bit-depth audio by sacrificing some precision while maintaining the original dynamics of the sound. In the picture below, an example between 16 bit and 24 bit depth is demonstrated. While 24 bit captures more data, it also consumes more data capacity. Therefore, in this method, I will test it out between 8 bit and 4 bit of data only.

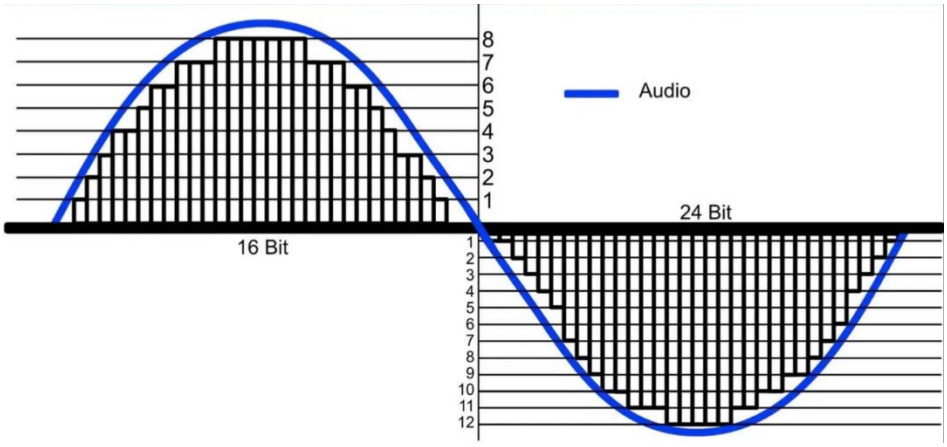


Figure 1: Example of Bit Depth (16 bit and 24 bit).

b) Convert to Mono:

Convert to mono method is used to convert stereo audio, which contains two separate channels (typically left and right), into a single-channel mono audio. This is useful for simplifying audio processing tasks, especially when working with machine learning models or algorithms that expect a single audio channel. In the provided code, the system takes the file path of an audio file as input. It uses the librosa.load() function, which loads the audio file and automatically converts it to mono by setting the mono=True parameter. This function averages the two stereo channels into one mono channel, effectively combining the left and right channels into a single channel of audio. The function then returns the mono audio signal along with the sample rate (sr). This method ensures that the output is a single, combined channel, which is often more computationally efficient and easier to process for many audio analysis tasks.

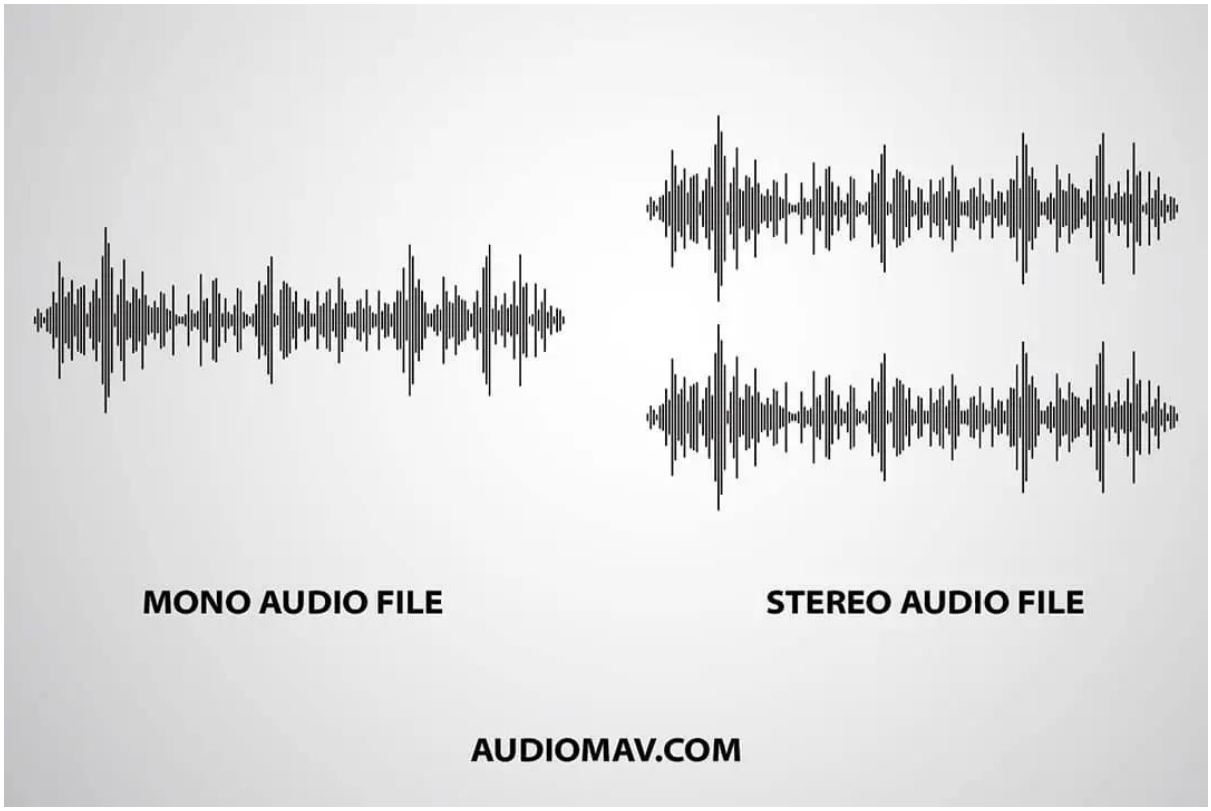


Figure 2: Mono and Stereo differences.

c) Trim silence:

The **trim silence** method is designed to remove silent portions of an audio file, effectively reducing its duration by cutting out sections where the audio signal is below a certain threshold of loudness. This is particularly useful for audio preprocessing, as it can help focus on the meaningful parts of the audio, such as speech or music, while removing unnecessary quiet segments. In the provided code, the function takes the file path of the audio file and an optional top_db parameter, which specifies the threshold in decibels below which audio is considered silent. By default, top_db is set to 20 dB. The function uses the librosa.load() function to load the audio file, then applies librosa.effects.trim() to remove the silent parts. This trimming operation compares the amplitude of the audio signal against the top_db threshold, and any audio segments that fall below this threshold are discarded. The method returns the trimmed audio signal along with its sample rate (sr). This approach is useful for audio processing tasks where silence does not contribute to the analysis, helping reduce file size and computational load by removing unnecessary parts of the audio.

d) Downsampling (Not implement):

Downsampling refers to the process of reducing the sample rate of an audio signal, which means decreasing the number of samples per second that represent the audio. This technique is commonly used to reduce the file size of audio data or to match the sample rate required by certain applications or devices. Downsampling is achieved by discarding or averaging the samples of the original high-resolution audio signal to produce a lower-resolution version. For instance, if an audio file is sampled at 48 kHz, downsampling might reduce the sample rate to 16 kHz, which lowers the overall data size and computational requirements, but it also results in a loss of high-frequency details in the audio. The trade-off involves balancing

file size and audio quality, where higher downsampling values lead to greater compression but also more noticeable degradation in sound quality.

e) Compression (Not implement):

Compression is a method used to reduce the size of an audio file by encoding the data in a way that removes redundant or unnecessary information while attempting to retain the most important aspects of the sound. There are two primary types of audio compression: lossless and lossy. Lossless compression methods, such as FLAC, preserve the original audio quality without any loss of data, meaning the audio can be perfectly restored when decompressed. On the other hand, lossy compression methods, such as MP3 or AAC, reduce file size by discarding parts of the audio signal that are less perceptible to the human ear, resulting in some loss of quality. Compression is essential in managing large audio datasets or streaming audio over the internet, as it helps minimize storage requirements and reduces transmission times. However, the effectiveness of compression depends on the algorithm used and the desired balance between file size and audio fidelity.

3) Yam-Net Model:

YAMNet (also known as Yet Another Mobile Network) is a pre trained deep learning model that is designed for audio event detection and classification. YAMNet is developed by Google and based on the MobileNetV1 architecture and uses audio features like log Mel spectrograms to recognize 521 sound classes from the Audio Set dataset such as speech, animal, environment noises. Because of its lightweight design, I have chosen this model for its efficient in mobile devices, followed by it sparse range of classes.

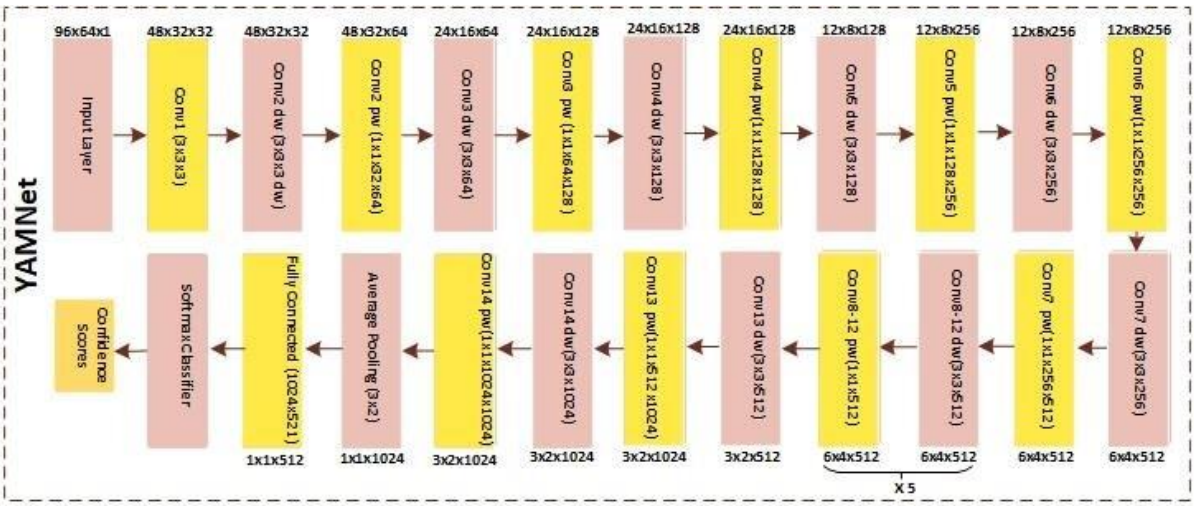


Figure 3: YAMNet architecture.

Structure of YAMNet model:

- Input Layer: Accepts log-Mel spectrogram with a shape of (96, 64) where 96 is time frame and 64 is the Mel frequency bands.
- Feature Extractor: A series of depth wise separable convolutional layers from the MobileNetV1 architecture. These layers are efficient and suitable for real-time applications, capturing the spatial and temporal patterns in the spectrogram.
- Global Average Pooling: Aggregates feature maps into a single vector, reducing the dimensionality while retaining the most important information.

- Full Connected Layer: Maps the extracted features to the output classes, producing the probability distribution over the 521 classes.
- Softmax Output: Provides class probabilities, enabling multi-label classification for overlapping sound events.

How does YAMNet work:

- Input representation: YAMNet processes audio as log-Mel spectrograms, which are compact representations of sound energy across time and frequency. The input to the model is typically a 1-second audio clip, sampled at 16 kHz, converted to a 64-band Mel spectrogram.
- Feature Extraction: The Mel spectrogram is fed into the model, where convolutional layers extract hierarchical features. These features represent the time-frequency patterns associated with different sound classes.
- Classification: YAMNet outputs a probability distribution over 521 sound event classes from the Audio Set ontology. Each probability represents the likelihood that a specific sound event is present in the input audio.
- Post Processing: Users can threshold or aggregate these probabilities over time to detect and classify sound events in longer audio clips.

C) Set up and Configuration:

1) Datasets and evaluation:

Based on my knowledge, each animal has its own sound features. While some of them have high frequency (bird, mouse, cat), there are others that have low frequency (cow, lion, tiger). Therefore, I will test each animal separately on the YAMNet model. For the evaluation, I will focus mostly on the model accuracy, which is measured by number of either correct prediction (on specific animal species) or animal recognitions only. Below is the list of animals and its expected prediction (other than Animal prediction):

- Aslan: Roaring cats (lions, tigers): Lion.
- Cat: Roaring cats (lions, tigers).
- Dog: Canidae, dogs, wolves.

2) Processing steps:

In general, the data will be trained with YAMNet as normal to see the standard accuracy that the model will acquire for that animal. All the data files are in .wav format. To preprocess the data, after loading data, I set the sample rate to 16 kHz. Then, I normalize the values between -1 and 1. The processed waveform will be sent as input into the model. Once getting the scores and predicted classes, I will retrieve the top class and top probabilities to add it into the result. When all data gets its result, the accuracy is calculated by the number of correctly predicted classes divided by the total number of prediction (equal to the total number of files loaded).

The data will be process separately with these methods:

- Reduce Bit Depth: Reduce the bit depth to 8 bit and 4 bit.

- Convert Stereo to Mono.
- Trim silence: Range from 0 to 80 dB with range 5. I believe that this is the range that most animal will occupy in.

To test the unitized capacity, I have created another file to inspect the current capacity and capacity after using each of these method and all three combined.

These are the variables that can be adjusted based on your desire for experiment.

- Audio directory path to your audio files (must be in .wav format).
- Desired class (refer to the class map of the model).
- Minimum and maximum range of Trim silence to cut (min_db and max_db).
- Distance between each cut.

3) Visualization:

These are the visualization that I will be used for displaying audio files and results. In this part, I will focus on mostly the waveform of the audio so I have commented all

a) Waveform:

The waveform visualization depicts the amplitude of the audio signal over time, providing a snapshot of its loudness dynamics. Peaks in the waveform indicate high-amplitude sections, which correspond to louder sounds, while troughs show quieter moments. This visualization is especially useful for identifying temporal features such as when certain events occur, sudden changes in volume, or periods of silence. For example, in speech, there might be some distinct pauses between words or sentences, whereas in music, rhythmic beats or sudden crescendos might stand out. Waveforms provide an intuitive way to assess the overall structure and loudness variation of the audio file.

b) Log Mel Spectrogram (optional, commented code):

The log Mel spectrogram transforms the audio signal into a time-frequency representation, showing how the intensity of frequencies evolves over time on a perceptual Mel scale. This visualization highlights both the pitch and timbre of the audio, with horizontal bands indicating dominant frequencies or tones. Darker regions signify areas of lower intensity, while brighter ones reflect high-energy frequencies. The log scaling emphasizes lower frequencies, making it easier to analyze speech patterns, musical notes, or bass-heavy components. By visualizing the time and frequency domains together, the spectrogram is particularly valuable for understanding complex audio features, such as phonemes in speech or chord progressions in music.

c) Frequency Spectrogram (optional, commented code):

The frequency spectrum offers a static view of the distribution of frequencies within the audio, showing the amplitude of each frequency component. By analysing the peaks, user can determine which frequencies dominate the signal, such as low frequencies for bass notes or high frequencies for treble and harmonics. This visualization is ideal for evaluating the tonal balance of the audio, whether it is bass-heavy, mid-range-focused, or treble-rich. It also reveals harmonic relationships, which are common in both speech and musical recordings. The frequency spectrum complements the other two visualizations by providing a clear, overall summary of the spectral content.

d) Line Graph:

To show the relationship between missing trimmed part with model accuracy, I use this graph to demonstrate it.

4) Capacity test:

To test how efficiency the reducing methods, I have included another code to test the capacity. The system will process the audio in directory /Sound/(animal_name) with each of the method (3 cases in total) and a case with all of them, followed by saving it in new directory /Sound/(animal_name)_method (BitDepth, S2M, TS, combined). These are the settings:

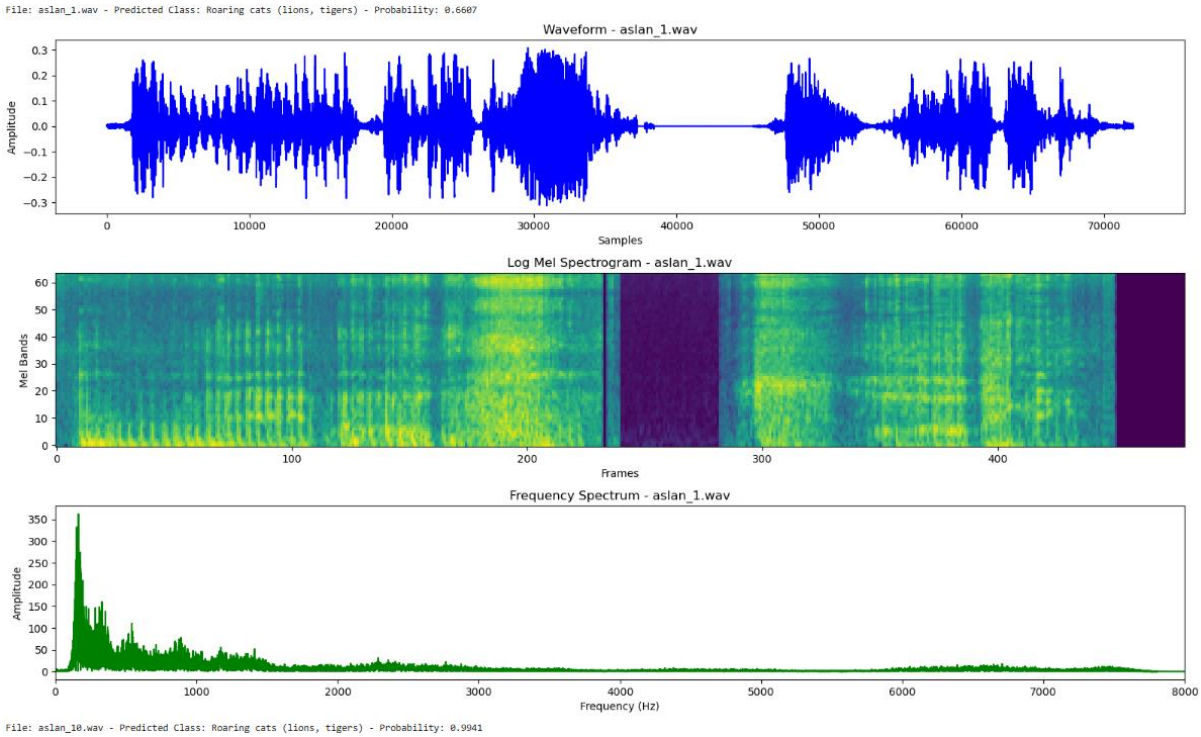
- BitDepth: Reduce to 8 bit.
- S2M: Default.
- TS: 20 dB for top_dB.

D) Result:

1) Aslan (Lion):

- Desired class: Roaring cats (lions, tigers); Animal.

The lion is a majestic big cat commonly referred to as the "king of the jungle," though it typically inhabits savannas, grasslands, and open woodlands rather than dense forests. Native to Africa and a small region in India, lions are social animals that live in groups called prides. In audio classification, the lion's roar is a key feature often analysed, characterized by its low-frequency rumble, high amplitude, and significant volume, capable of traveling up to 8 kilometres. These acoustic features make lion roars distinct and are crucial in identifying the species in bioacoustics studies, enabling researchers to monitor populations and behaviours in their natural habitat. For this animal, I have 45 audio samples for classification. Below is an example and visualization of audio file when predicting with the model.




```
Model Accuracy of: Sound/Aslan
88.88888888888889
```

Figure 4: Example and base result of Aslan.

In this example, the roar of lion is divided in two stages. In each part, the amplitude reaches maximum at nearly 0.3 in the first one and 0.25 in the second one. While the duration is shorter in the second stage, the first stage shows more intense in Mel bands. However, in both stages, the frequency of lion mostly focuses on below 1000.

To inspect further into the result, I will use each of three methods mentioned above separately to inspect the Impact it has on the size and Quality of Audio:

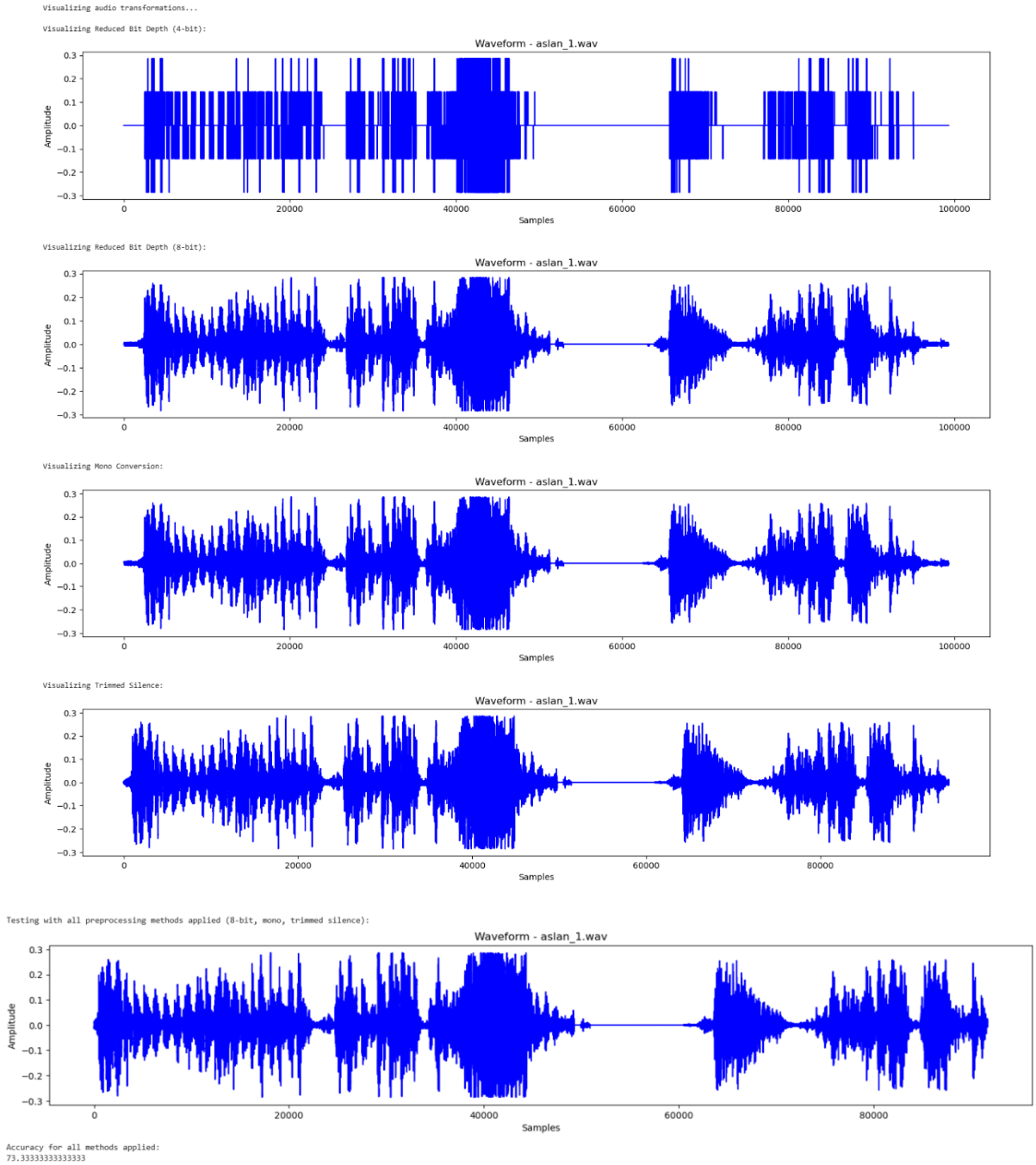
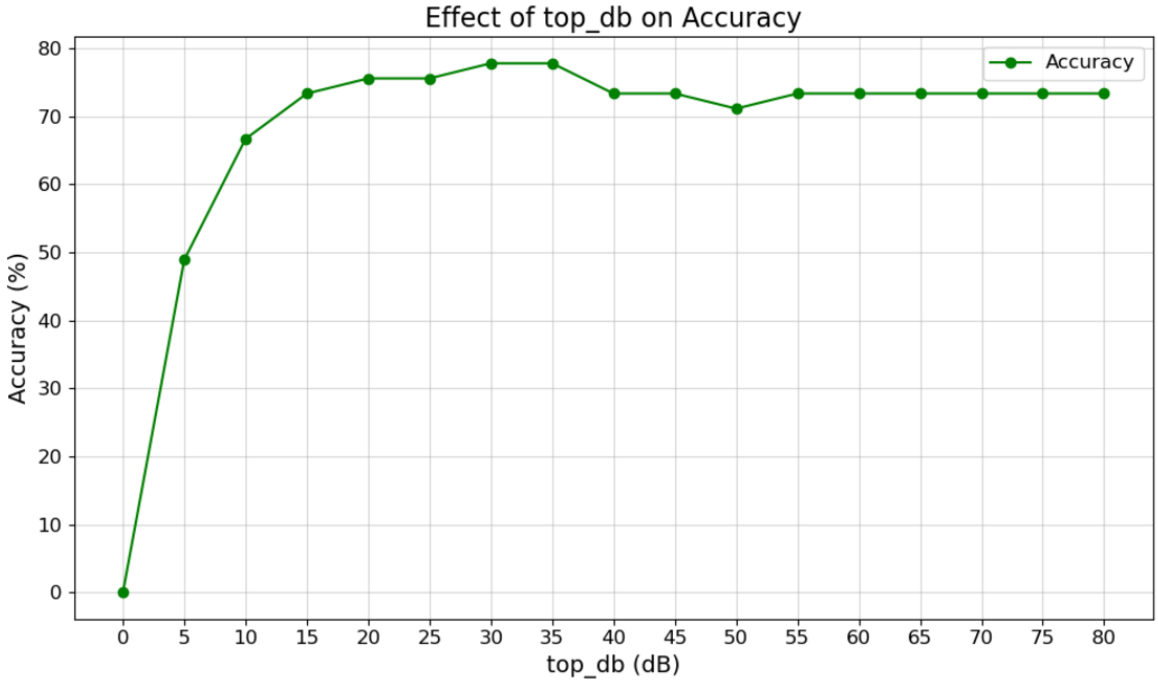


Figure 5: A significant example of using Reduced Bit Depth (4 bit and 8 bit), convert Stereo to Mono, Trim Silence (20 dB) and combine all methods.

When using 4 bit depth, the data seems more general and harder to recognize the patterns. Comparing with 8 bit depth, the small detail of 8 bit is larger than 4 bit, followed by some higher amplitude particles (for example at timeline 40000). Furthermore, the first part of 4 bit is considered silent in 4 bit, which might affect the accuracy later in the prediction. In mono conversion, there is not much different from the standard except for increases amplitude. Next, in trimmed silence, the parts that are 20 dB or below are removed, result in louder starter in sound as can be seen in the picture. Finally, when combining all methods together, the audio has even larger starter and ending in audio.

```
Testing with trimmed silence (top_db=50):
Accuracy for top_db=50: 71.11%

Testing with trimmed silence (top_db=55):
Accuracy for top_db=55: 73.33%

Testing with trimmed silence (top_db=60):
Accuracy for top_db=60: 73.33%

Testing with trimmed silence (top_db=65):
Accuracy for top_db=65: 73.33%

Testing with trimmed silence (top_db=70):
Accuracy for top_db=70: 73.33%

Testing with trimmed silence (top_db=75):
Accuracy for top_db=75: 73.33%

Testing with trimmed silence (top_db=80):
Accuracy for top_db=80: 73.33%
```




```
Testing with reduced bit depth (4-bit):
Accuracy for 4-bit reduction:
20.0

Testing with reduced bit depth (8-bit):
Accuracy for 8-bit reduction:
82.22222222222221

Testing with mono conversion:
Accuracy for mono conversion:
73.33333333333333

Testing with trimmed silence:
Accuracy for silence trimming:
75.55555555555556
```

```
Directory sizes (in MB):
Base Directory: 5.01 MB
After Reduce Bit Depth: 2.55 MB
After Stereo to Mono: 2.55 MB
After Trim Silence: 2.25 MB
After Combined Preprocessing: 2.25 MB

Directory Size Comparison:
                    Method  Size (MB)
0           Base Directory   5.014977
1         Reduce Bit Depth   2.548014
2           Stereo to Mono   2.548014
3             Trim Silence   2.245276
4  Combined Preprocessing    2.245276
```

Figure 6: Result of testing.

In the result, as can be seen from the pictures, while 4 bit reduces the accuracy to 20%, other methods only reduce to 82% and 73%. When testing accuracy with different value of dB, removing only completely 0 dB decreases the accuracy to 0%, while increasing the dB might improve the accuracy and let it remain stable at 73%. This indicates that removing only 0 dB part might increases the weight of irrelevant features, altering the prediction. My claim is

further proved when combining all methods, result in stable accuracy at 73% as well. In terms of size, all methods save the capacity by nearly half, except for trim silence with more than half. When combining all methods, trim silence decides the final result.

2) Cat:

- Desired class: Cat; Roaring cats (lion, tigers) (optional); Domestic Animals, pets; Animals

Unlike Aslan (lion), the audio files of cats are quite interesting. To be specific, this is mostly considered a domestic animal, raising by human. Therefore, the sound of cats is affected by their living environment, changing the frequency significantly but mostly have higher frequency than lions. As a result, the prediction is more likely to be classified as human activities like speech, crying, sobbing, etc. With 200 audio samples, this is what I have found:

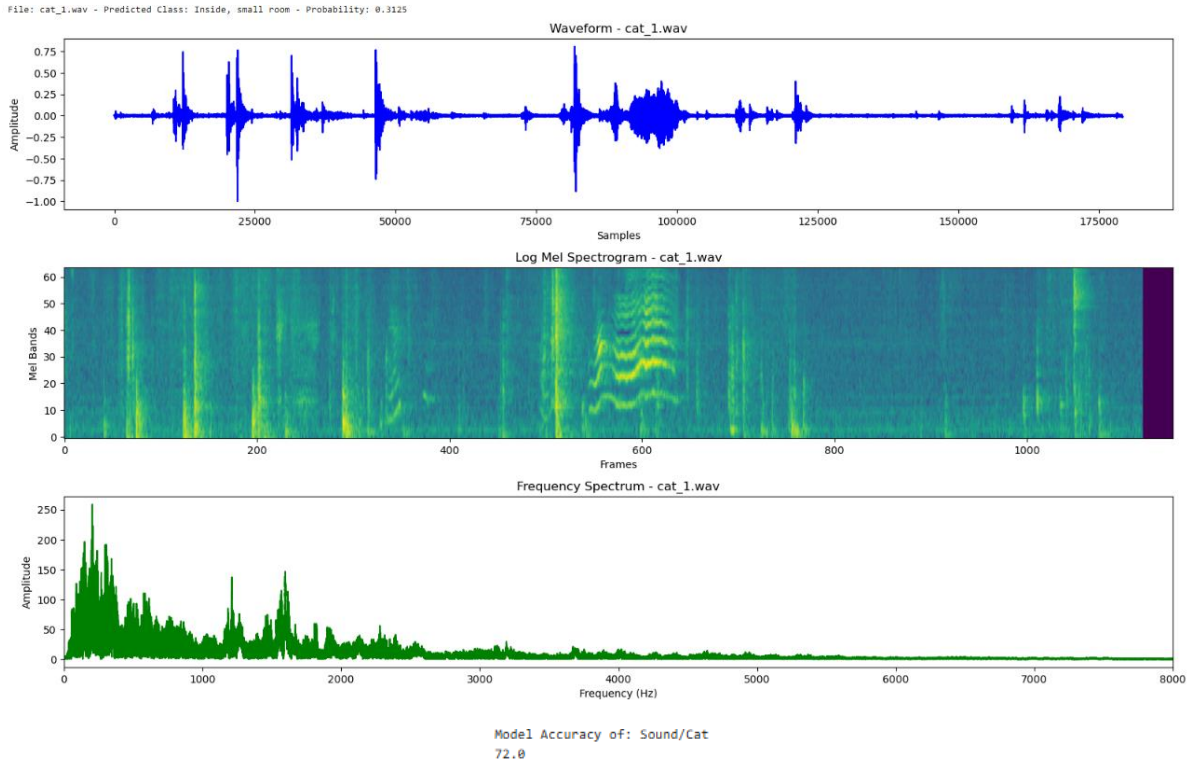


Figure 7: Example and base result of Cat.

As can be seen from the graph, unlike lion, cat has smaller loudness. Moreover, the sound only focusses on a few small parts of the audio. Moving to the Frequency spectrum, the distribution of sound frequency spreads more than lions with more patterns over 3000 Hz.

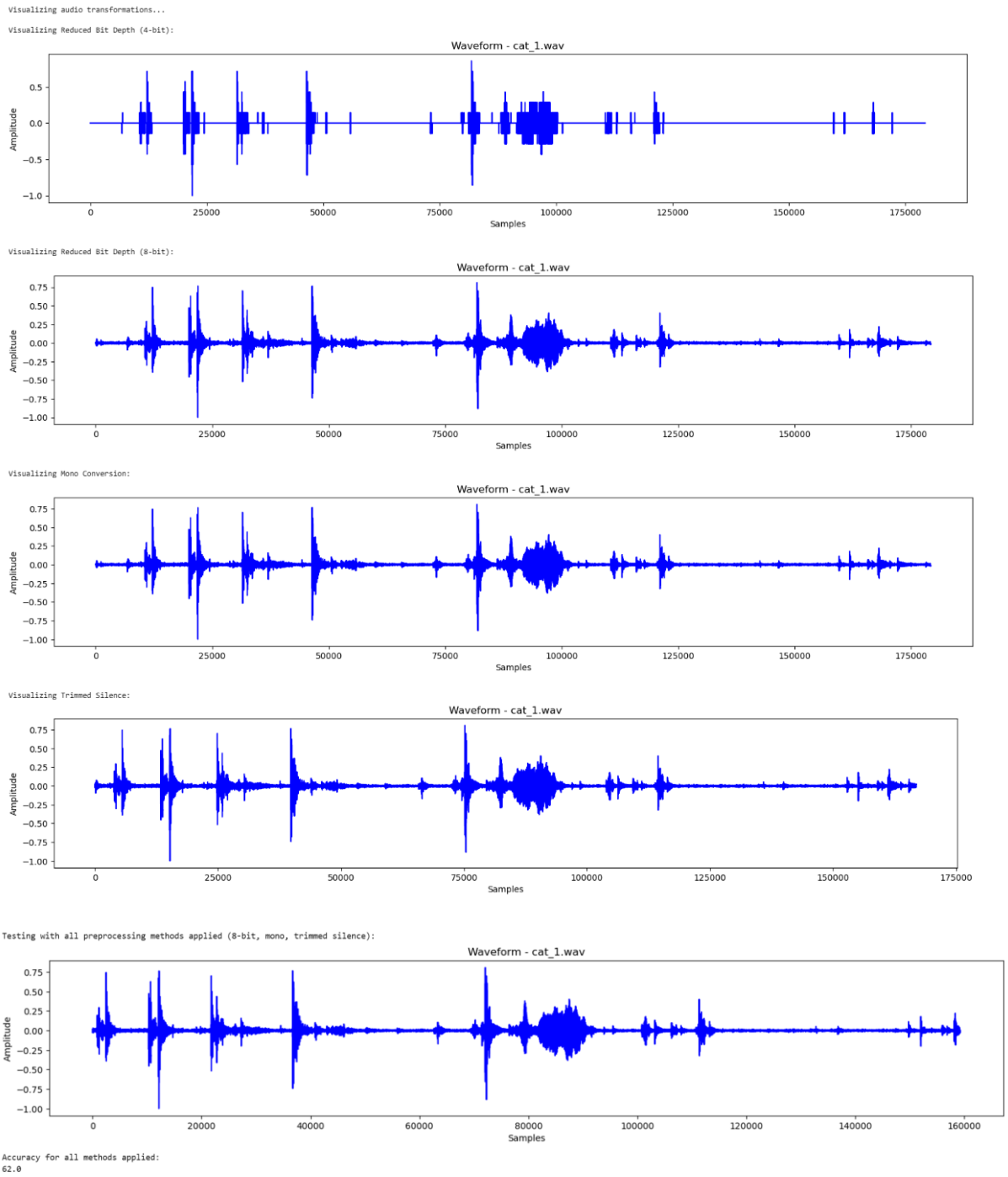
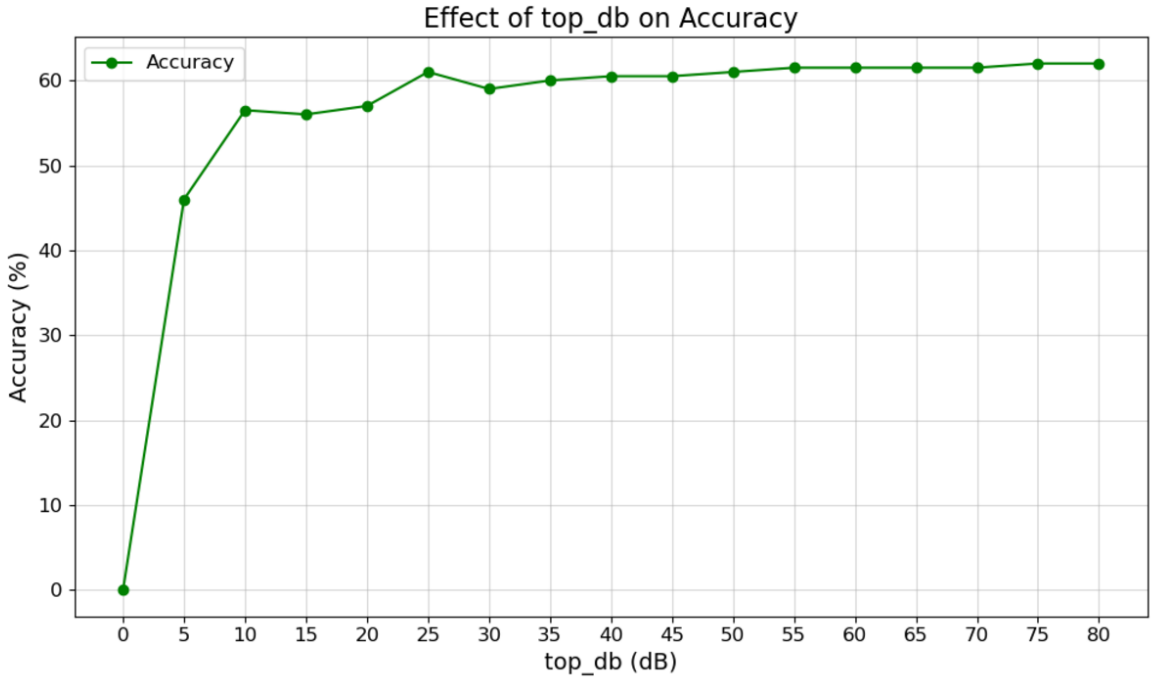


Figure 8: A significant example of using Reduced Bit Depth (4 bit and 8 bit), convert Stereo to Mono, Trim Silence (20 dB) and combine all methods.

In this animal, because the audio of cat contains audio with low amplitude but high frequency, the 4 bit method shows not much difference comparing to 8 bit. However, while mono conversion increases the amplitude of the sound, trimming silence also removes the silence part and decrease the size. To be specific, when combine all methods together, the duration decreases from nearly 18000 to 16000, followed by louder starter than standard one. Nevertheless, the features of cat audio remain unchanged. Moving on, let's dive into the result.

```
Testing with trimmed silence (top_db=60):
Accuracy for top_db=60: 61.50%

Testing with trimmed silence (top_db=65):
Accuracy for top_db=65: 61.50%

Testing with trimmed silence (top_db=70):
Accuracy for top_db=70: 61.50%

Testing with trimmed silence (top_db=75):
Accuracy for top_db=75: 62.00%

Testing with trimmed silence (top_db=80):
Accuracy for top_db=80: 62.00%
```




```
Testing with reduced bit depth (4-bit):
Accuracy for 4-bit reduction:
30.5

Testing with reduced bit depth (8-bit):
Accuracy for 8-bit reduction:
67.5

Testing with mono conversion:
Accuracy for mono conversion:
68.5

Testing with trimmed silence:
Accuracy for silence trimming:
66.5
```

```
Directory sizes (in MB):
Base Directory: 41.67 MB
After Reduce Bit Depth: 21.21 MB
After Stereo to Mono: 21.21 MB
After Trim Silence: 17.92 MB
After Combined Preprocessing: 17.92 MB

Directory Size Comparison:
                    Method  Size (MB)
0           Base Directory  41.666146
1         Reduce Bit Depth  21.212008
2           Stereo to Mono  21.212008
3             Trim Silence  17.918062
4  Combined Preprocessing   17.918062
```

Figure 9: The result of Cat.

In this result, using reduced bit depth method decreases the accuracy by 40%. Meanwhile, for other methods, the accuracy only goes down to 67% to 68% before reaching the lowest at 62% for all combined. For file capacity, similarly, using reduce bit depth and mono conversion will nearly halve the size, while using trim silence and combined preprocessing will reduce the file more than half as a result of cutting duration.

3) Dog:

- Desired class: Dog; Crunch; Domestic animals, pets; Animals.

Moving on to Dog, this is another domestic animal. However, unlike cat, dogs have low focuses on low frequency like lions but higher amplitude. With 100 files, I have inspected, and these are my conclusions. In the waveform visualization, dog audio contains short duration but high amplitude (around 0.5, higher than lion). However, in the frequency spectrum, dog concentrates mostly between 200 to nearly 2000 Hz. Therefore, removing the part between 100 to 200 Hz does not affect much the feature of dog. When using the Bandstop Filtering, while 1200 to 1500 and 1700 to 2000 are considered the noise of data, the range 700 Hz to 900 Hz is considered the most important feature of the data.

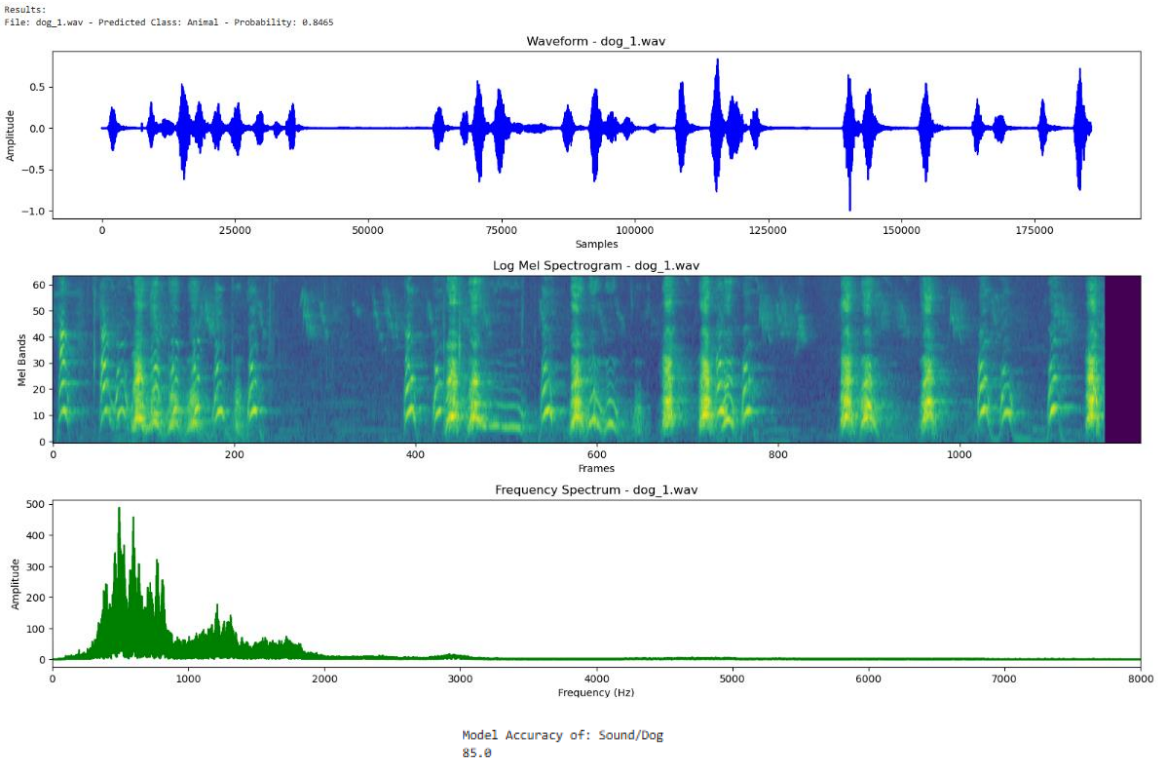


Figure 10: Example of Audio visualization of dog.

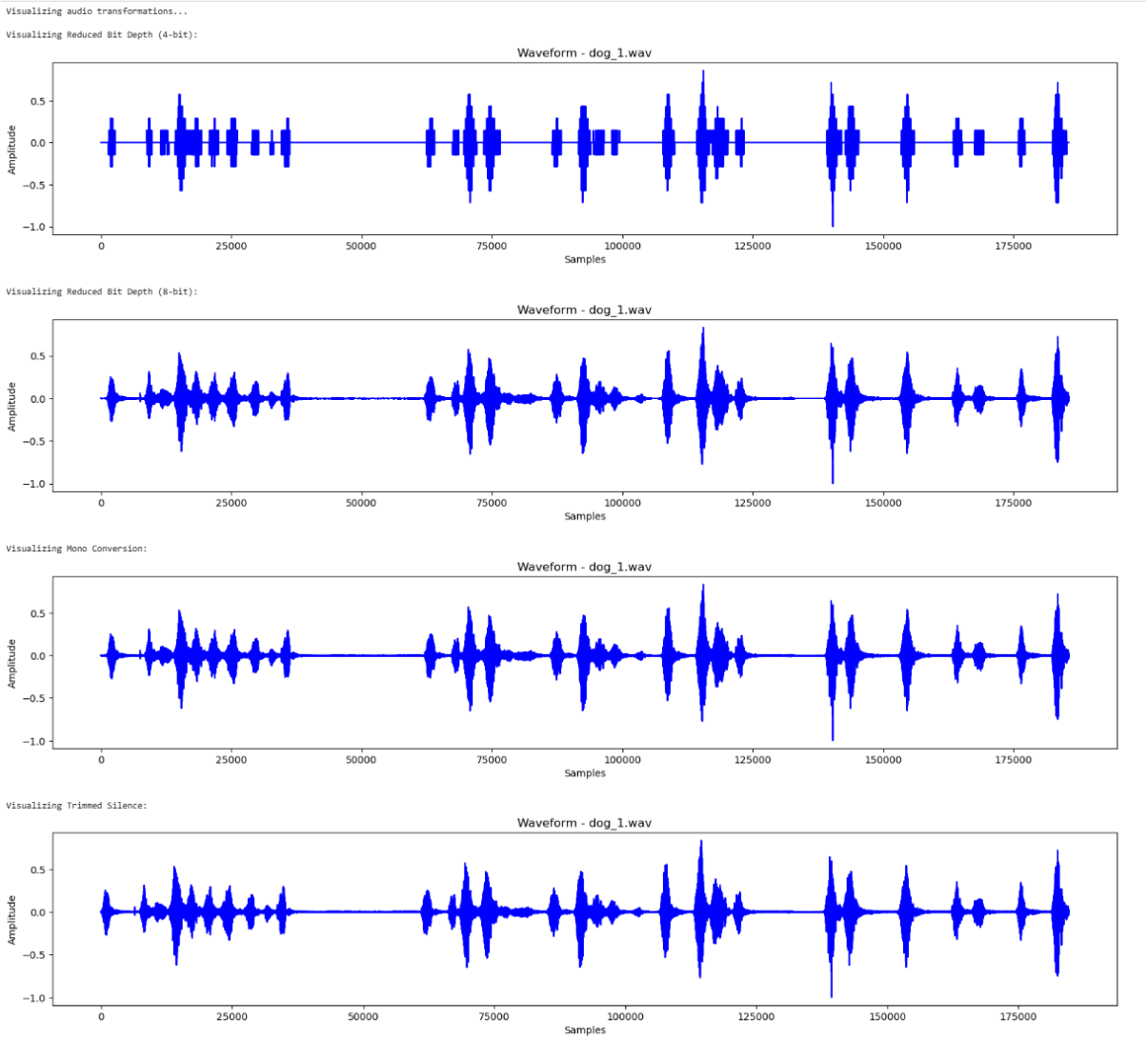
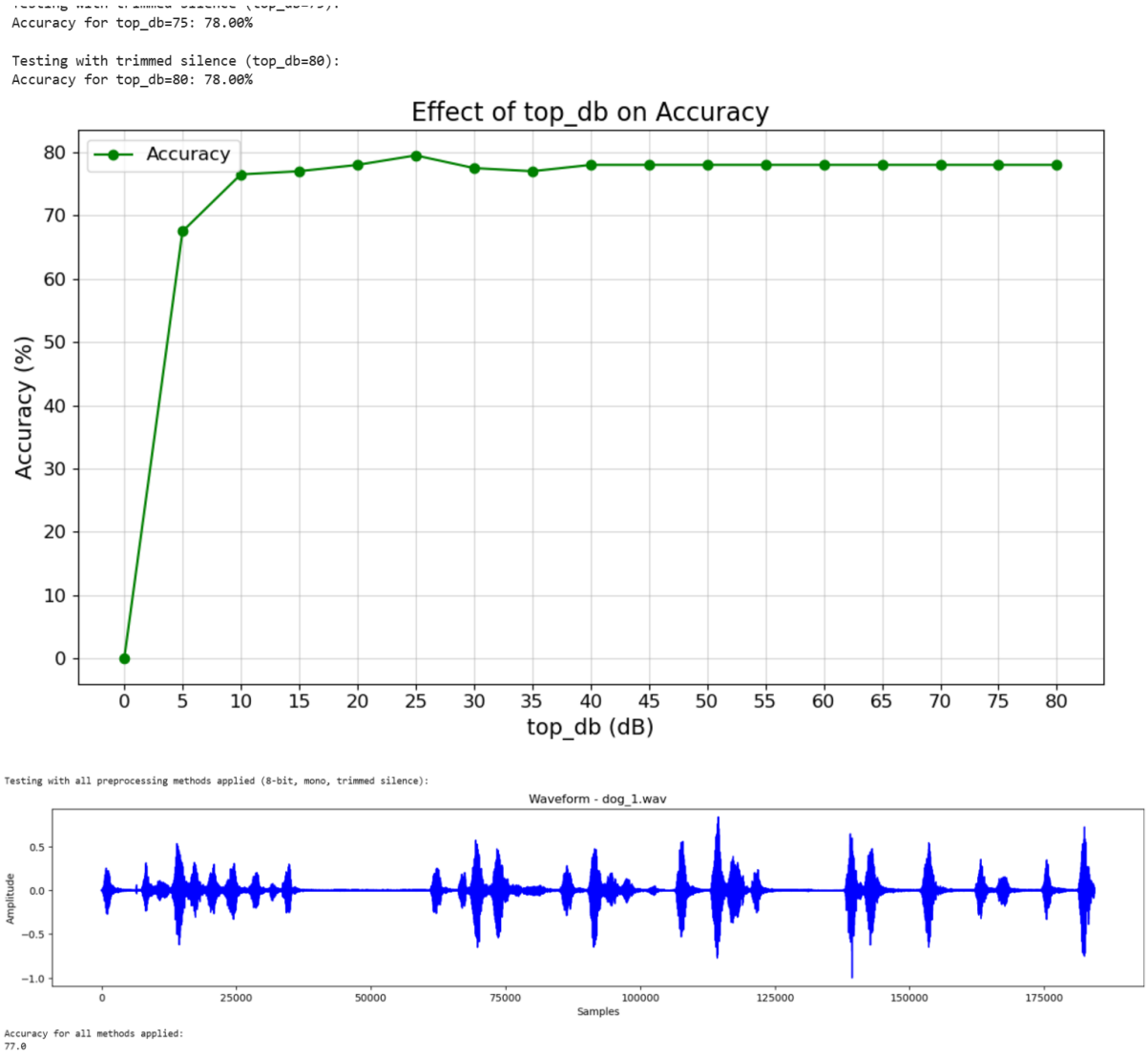


Figure 11: A significant example of using Reduced Bit Depth (4 bit and 8 bit), convert Stereo to Mono, Trim Silence (20 dB) and combine all methods.

Moving on to dogs, as can be seen from the graph, the differences between reducing to 8 bit, 4 bit and standard one is really small and can't be witnessed in normal observation. To be specific, reducing to 4 bit shows a little more general in record than standard one. Furthermore, in mono conversion, the changes are even harder to spot. However, the trim silence is shown really effective in reducing the duration, result in louder starter and rest part. This will help the model learn the data more effective when detecting the audio parts of animals that are really small.




```
Testing with reduced bit depth (4-bit):
Accuracy for 4-bit reduction:
32.5

Testing with reduced bit depth (8-bit):
Accuracy for 8-bit reduction:
78.5

Testing with mono conversion:
Accuracy for mono conversion:
78.0

Testing with trimmed silence:
Accuracy for silence trimming:
78.0
```

```
Directory sizes (in MB):
Base Directory: 23.78 MB
After Reduce Bit Depth: 12.95 MB
After Stereo to Mono: 12.95 MB
After Trim Silence: 11.40 MB
After Combined Preprocessing: 11.40 MB

Directory Size Comparison:
                    Method  Size (MB)
0           Base Directory  23.777458
1         Reduce Bit Depth  12.947412
2           Stereo to Mono  12.947412
3             Trim Silence  11.399631
4  Combined Preprocessing   11.399631
```

Figure 12: Result of using all those methods.

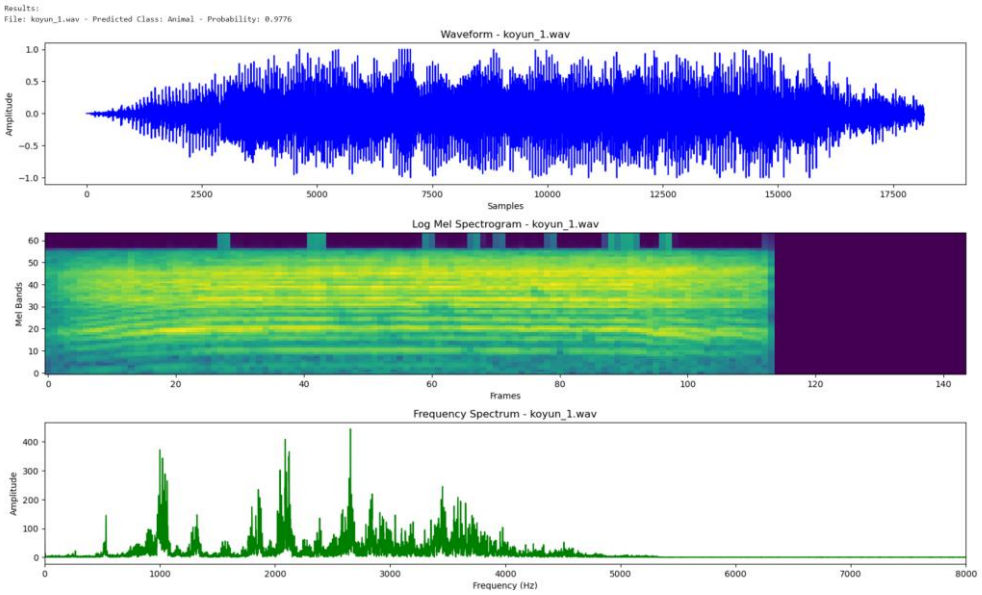
After testing all methods, I can conclude that although using all methods combined will reduce the size by more than half, the accuracy is also reduced significantly from 85 to 77%. However, the difference between combining all methods and other methods separately is only 1%. Therefore, it is recommended for using all methods together.

For the later animals, I will attach the result here and conclude my finding only.

4) Koyun (Sheep):

- Desired class: Sheep, Animals.

For ruminants, sheep is my chosen among all of them. Using 40 samples, I have observed some interesting information. In waveform, the sound of sheep is a long with high frequency and amplitude. Specifically, while the amplitude reaches its peak at nearly 1, the frequency spreads from 1000 to 4000 Hz. Therefore, the Bandstop filter only affect the frequency above 1000. As a result, the accuracy drops significantly to 65% when removing the 1000 to 1100 Hz patterns. For other parts, there is not much change in accuracy from 4600 Hz onward.




```
Model Accuracy of: Sound/Koyun
85.0
```

Figure 13: Example of Sheep.

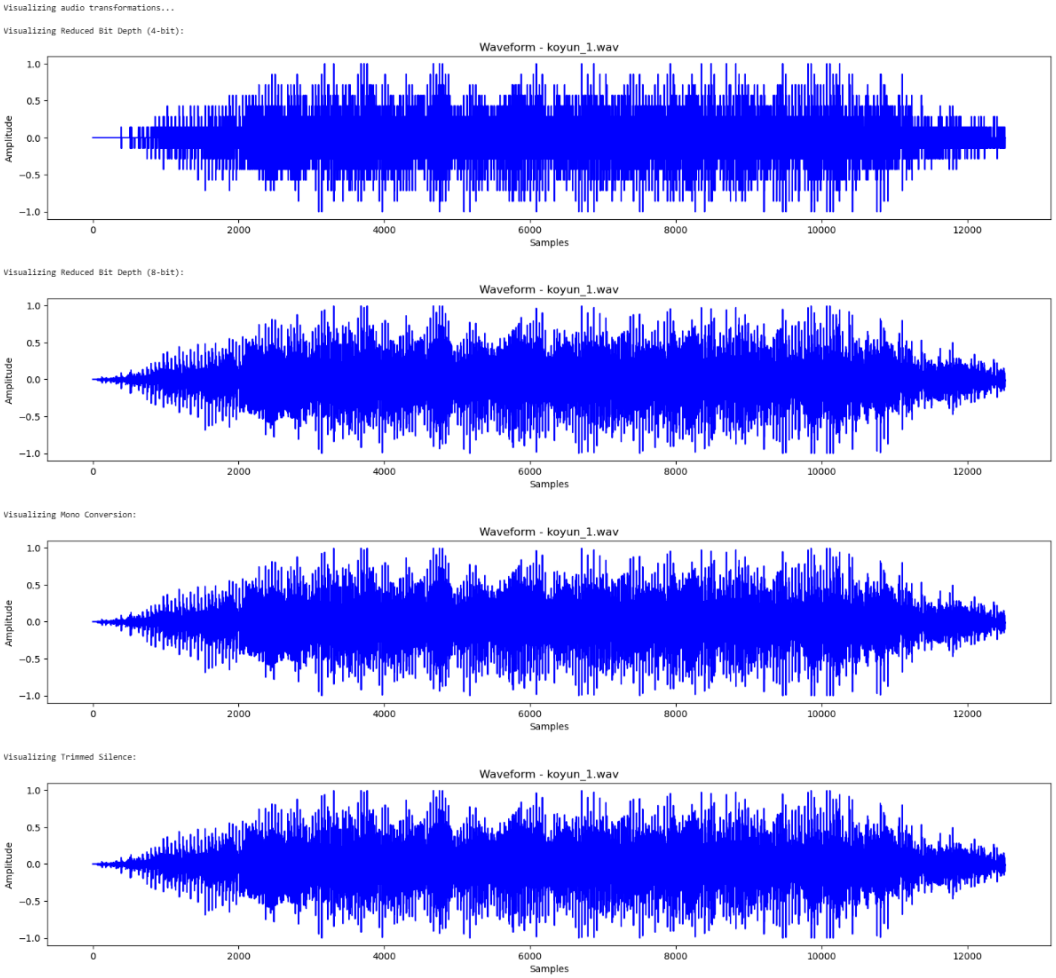
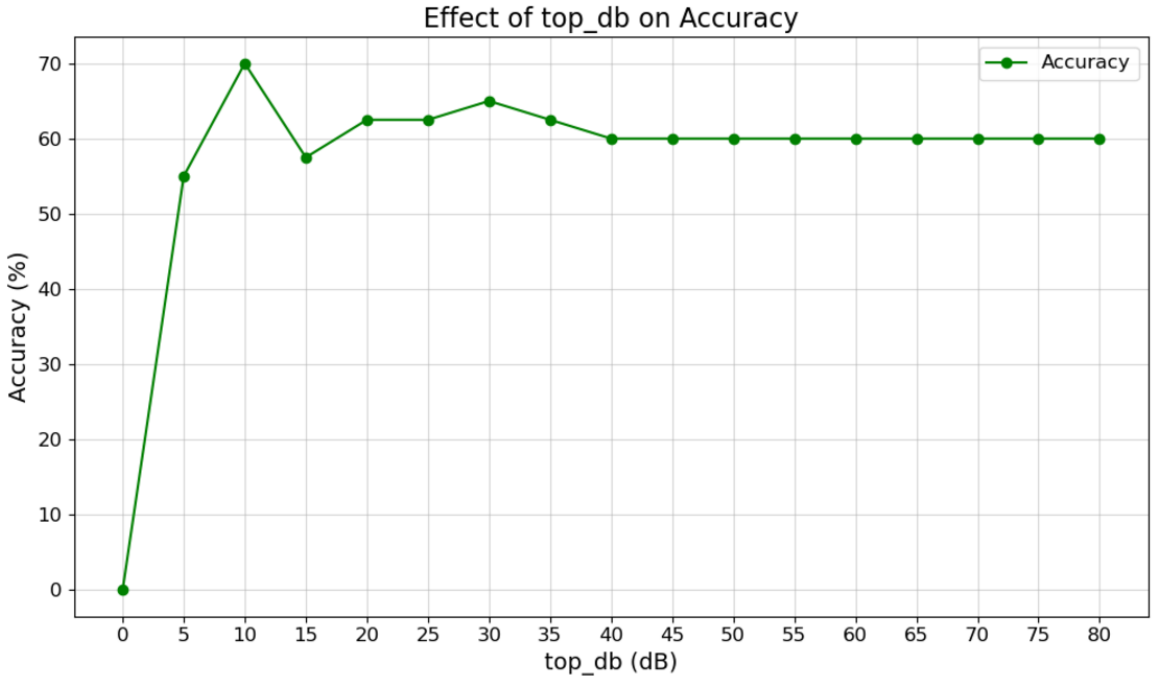
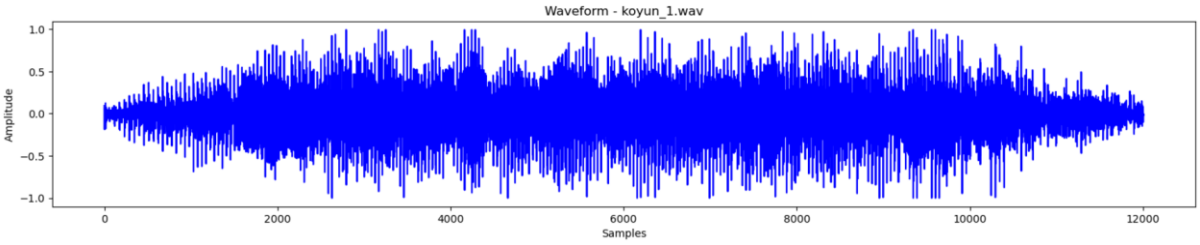


Figure 14: A significant example of using Reduced Bit Depth (4 bit and 8 bit), convert Stereo to Mono, Trim Silence (20 dB) and combine all methods.

```
Testing with trimmed silence (top_db=60):
Accuracy for top_db=60: 60.00%

Testing with trimmed silence (top_db=65):
Accuracy for top_db=65: 60.00%

Testing with trimmed silence (top_db=70):
Accuracy for top_db=70: 60.00%

Testing with trimmed silence (top_db=75):
Accuracy for top_db=75: 60.00%

Testing with trimmed silence (top_db=80):
Accuracy for top_db=80: 60.00%
```




```
Testing with all preprocessing methods applied (8-bit, mono, trimmed silence):
```




```
Accuracy for all methods applied:
57.49999999999999
```

```
Testing with reduced bit depth (4-bit):
Accuracy for 4-bit reduction:
40.0

Testing with reduced bit depth (8-bit):
Accuracy for 8-bit reduction:
67.5

Testing with mono conversion:
Accuracy for mono conversion:
60.0

Testing with trimmed silence:
Accuracy for silence trimming:
62.5
```

```
Directory sizes (in MB):
Base Directory: 2.74 MB
After Reduce Bit Depth: 1.46 MB
After Stereo to Mono: 1.46 MB
After Trim Silence: 1.30 MB
After Combined Preprocessing: 1.30 MB

Directory Size Comparison:
                   Method  Size (MB)
0          Base Directory   2.744769
1        Reduce Bit Depth   1.456649
2          Stereo to Mono   1.456649
3            Trim Silence   1.302767
4  Combined Preprocessing   1.302767
```

Figure 15: Result of testing the methods.

5) Kurbaga (Frog):

- Desired class: Frog; Animal.

Moving on to the wild animals, Frog is one of the animals that has sound frequency focuses on medium but remains high amplitude. To digest deeper, while the amplitude keeps unchanged at nearly 1, the distribution of frequency only in 500 to 2000 Hz. Therefore, when using Bandstop Filter, the patterns between 100 to 200 Hz drops around 10% of accuracy. However, some small details like 1100 – 1200, 1300 -1400, 3500 – 3600, etc increases the accuracy to even 94%, making these are also important noise that needs to be concerns. Along with waveform visualization, the noise can be alternated with features. However, the accuracy still remains stable from 3000 onward.

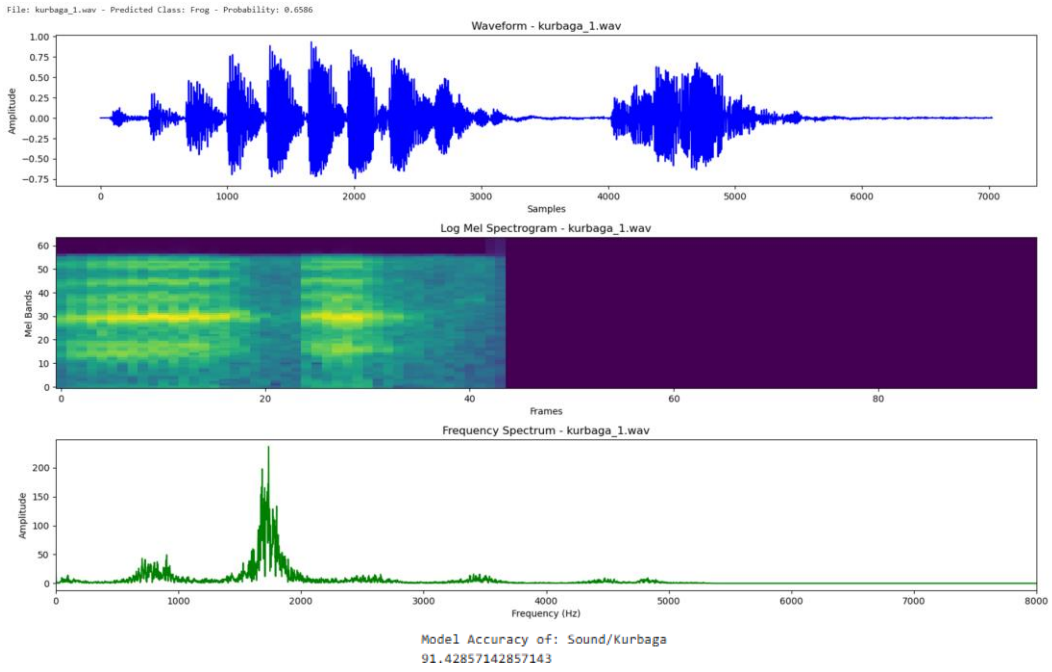


Figure 16: Example of Frog.

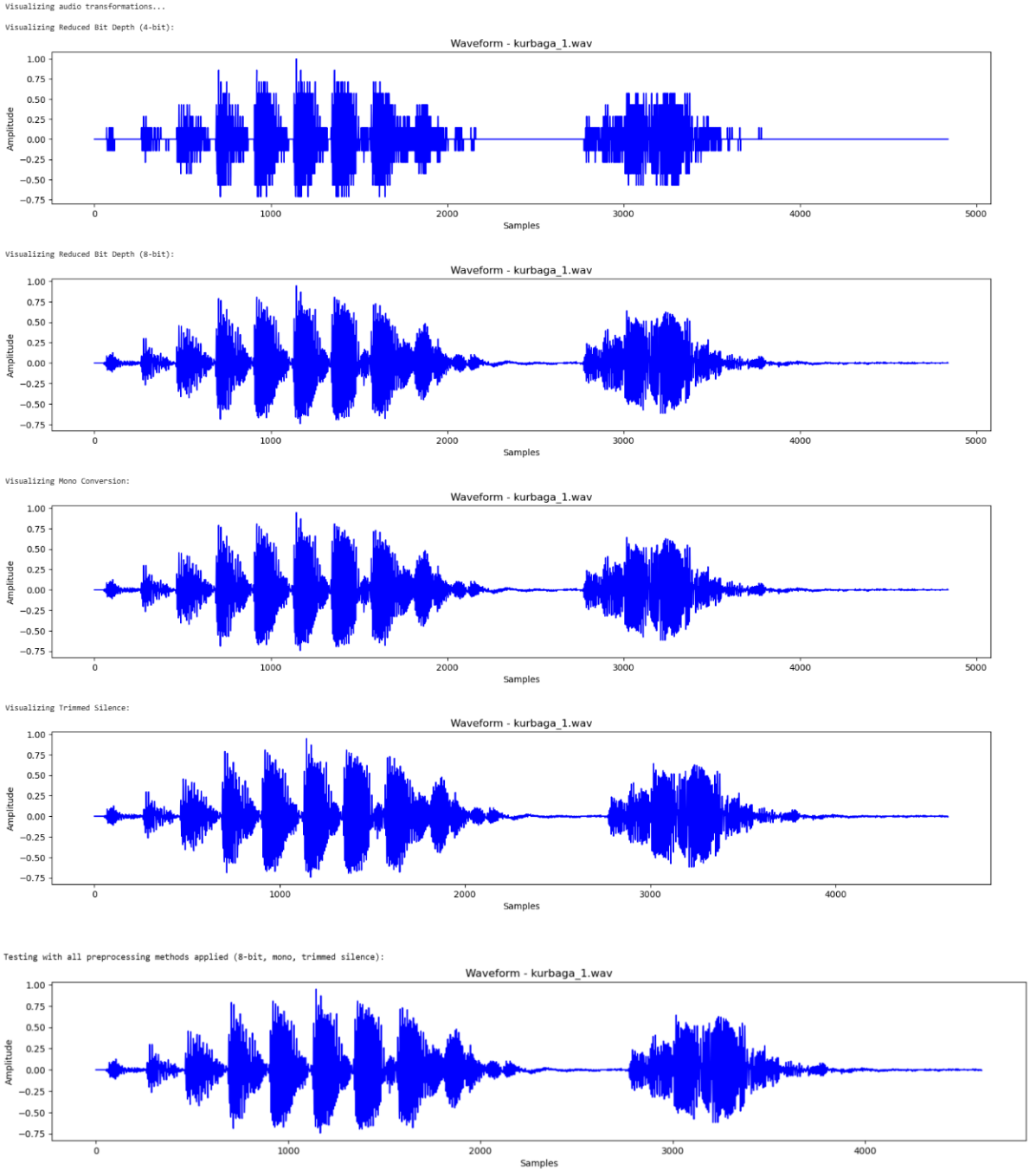
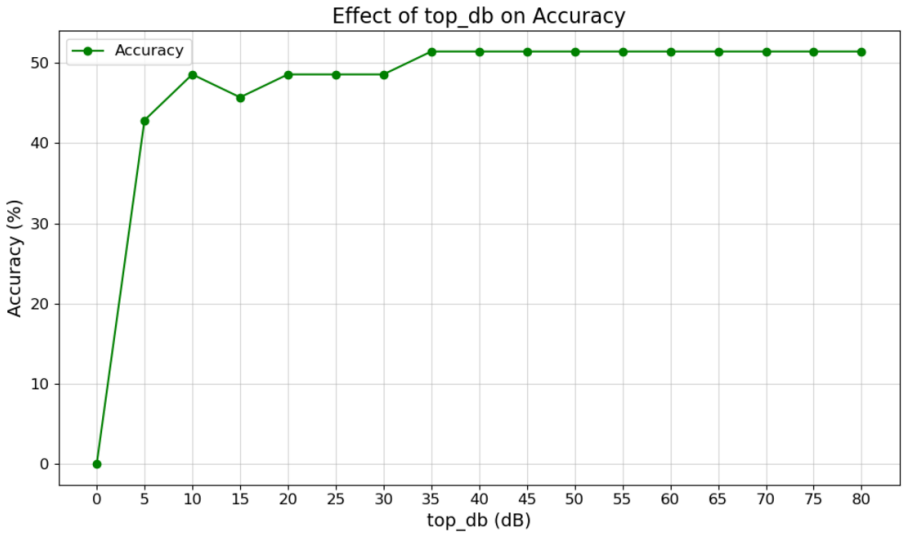


Figure 17: A significant example of using Reduced Bit Depth (4 bit and 8 bit), convert Stereo to Mono, Trim Silence (20 dB) and combine all methods.

```
Testing with trimmed silence (top_db=60):
Accuracy for top_db=60: 51.43%

Testing with trimmed silence (top_db=65):
Accuracy for top_db=65: 51.43%

Testing with trimmed silence (top_db=70):
Accuracy for top_db=70: 51.43%

Testing with trimmed silence (top_db=75):
Accuracy for top_db=75: 51.43%

Testing with trimmed silence (top_db=80):
Accuracy for top_db=80: 51.43%
```




```
Testing with reduced bit depth (4-bit):
Accuracy for 4-bit reduction:
14.285714285714285

Testing with reduced bit depth (8-bit):
Accuracy for 8-bit reduction:
45.714285714285715

Testing with mono conversion:
Accuracy for mono conversion:
51.42857142857142

Testing with trimmed silence:
Accuracy for silence trimming:
48.57142857142857
```

```
Directory sizes (in MB):
Base Directory: 1.66 MB
After Reduce Bit Depth: 1.14 MB
After Stereo to Mono: 1.14 MB
After Trim Silence: 1.01 MB
After Combined Preprocessing: 1.01 MB

Directory Size Comparison:
                   Method  Size (MB)
0          Base Directory   1.655310
1       Reduce Bit Depth    1.135732
2         Stereo to Mono    1.135732
3           Trim Silence    1.009672
4  Combined Preprocessing   1.009672
```

Figure 18: Result of testing methods.

6) Result summary:

Below is my table displaying the result from all above (note that Acc: %):

Animals (Audio size in Mb)	Accuracy	Reduce Bit Depth (4 bit) accuracy	Reduce Bit Depth (8 bit)		Stereo to Mono		Trim silence (all best around 20 dB)		Combined	
			Acc	Size	Acc	Size	Acc (avg)	Size	Acc	Size
Lion (5.01)	88.89%	20%	82.22	2.55	73.33	2.55	75.5	2.45	73.33	2.45
Cat (41.66)	72%	30.5%	67.5	21.2	68.5	21.2	62	17.91	62	17.91
Dog (23.77)	85%	32.5%	78.5	12.9	78	12.9	80	11.39	77	11.4
Sheep (2.74)	85%	40%	67.5	1.45	60	1.45	70	1.3	57.45	1.3
Frog (1.65)	91.43%	14.3%	45.71	1.13	51.43	1.13	48.57	1	45.71	1

E) Conclusions:

After conducting experiments in those audios of animal, I found that Trimming silence shows the most effective result in size reduction. However, in terms of accuracy, the reduction to 8 Bit Depth shows the highest accuracy. To be specific, while reducing Bit Depth and converting Stereo to Mono utilize nearly half of the size, trimming silence saves even more than half of the capacity. While in most animals, reducing Bit Depth shows the highest accuracy, trimming silence outstanding Bit Depth Reduction in dogs.

In short, each method is required for different purposes while reducing Bit Depth (specifically 8 bit) is used for the sound with high amplification like lions, tigers; trimming silence focuses more on medium amplification and consistent value. However, the stereo to mono conversion mostly concentrates on long and repeated part of sound. Furthermore, the trimming silence should be optimized for top_dB with value 20 or 25. This ensures that the size is reduced but the weight of irrelevant details is not increases too high.

F) Guide for Inspect the code and Reference:

1) Guide to test the code:

- Installed required library as listed in the code.
- Impact of Audio test code: Adjust the file directory and testing range for top dB to trim and run the code.
- Capacity check: Put the original audio directory and newly desired directory for modified audio files and run the code.

2) Reference:

- Link 1.
- Data Link.
- Link 2.
- Link 3.