



AUDIO COMPRESSION

USING DISCRETE COSINE TRANSFORM COMPRESSION
AND DYNAMIC SAMPLE ELIMINATION

AV331

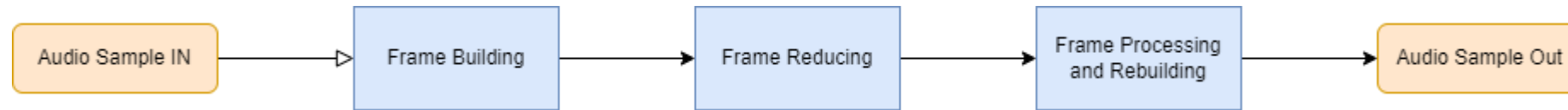
SC19B129

Arosish Priyadarshan

INTRODUCTION

- Audio compression is a well-studied area of digital signal processing and psychoacoustics. The study of audio compression is concerned with reducing the physical size of a signal while retaining the perceived integrity of the signal.
- This project makes use of two compression methods: “Discrete Cosine Transform Compression” and “Dynamic Sample Elimination ” a method to detect and remove irrelevant samples of data by applying a light variation of masking.
- The model built for this project is lossy audio compression.

ALGORITHM



The model is based on three major steps, i.e.

1. Building of Frames
2. Reducing these Frames
3. Reprocessing of these Frame of get the final Output.

PARAMETERS

Four Factors are used in the model to control the output.

1. Scaling Factor (DSE Factor)
2. Cast Factor (DCT Factor)
3. Frame Length (in seconds)
4. Overlap Ratio

APPROACH: (DISCRETE COSINE TRANSFORM COMPRESSION)

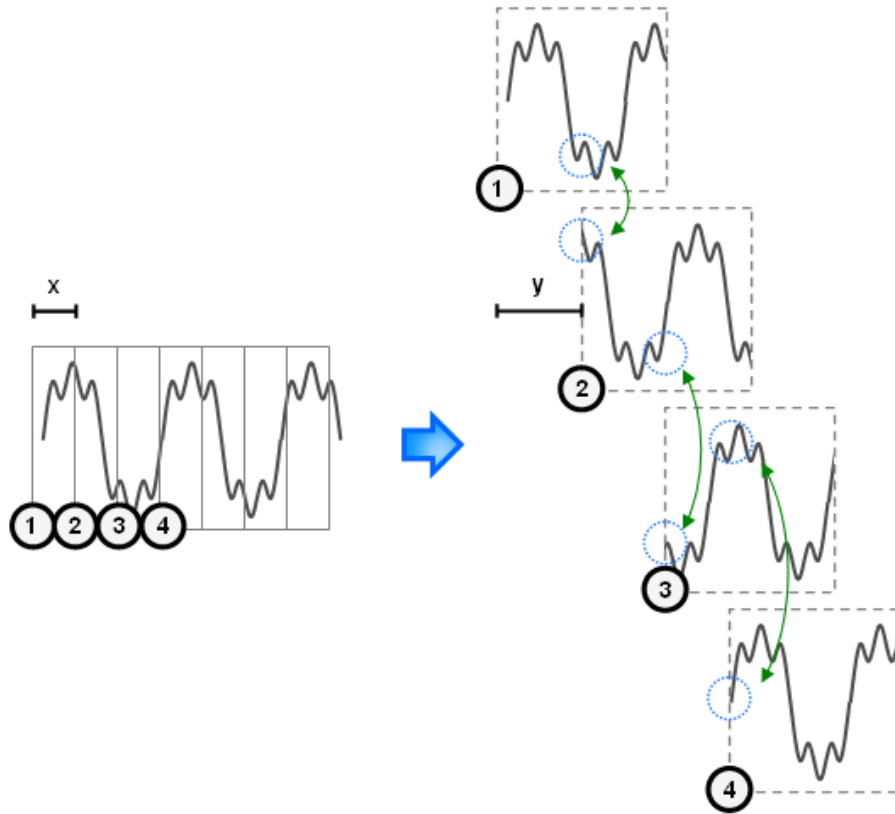
- This method of compression is commonly used in a large majority of high-grade compression algorithms such as MPEG due to low computational cost and high performance of the DCTC algorithm.
- The method works by taking a one-dimensional, long audio vector, the audio file is read by MATLAB. The audio file is processed compressed by the chosen **CAST**. The application of the discrete cosine transform is applied to the frames of the recording which are simply a region of the recording.

$$\text{Sample Factor} = \text{Frame length} / \text{Cast}$$

$$\text{Total Number of Frames} = \text{Recording length} / \text{Frame length}$$

- By applying this to every frame the signal frequency range can be reduced by the ratio of the **CAST** value.
- After the application of DCT to each frame of the audio signal, the signal is also down sampled by the **CAST** amount to normalize the number of samples to the sampling frequency in accordance with the original signal.

DYNAMIC SAMPLE ELIMINATION



- The idea behind this method is roughly based around the application of *temporal masking* and *hearing thresholds*.
- Algorithm works by splitting the audio signal into frames, with each frame having a frame length in seconds as **Frame Length (Seconds)** and **Overlap Ratio** which is a value between 0 and 1, indicating the amount of “frame shifting” done for a frame based on the frame’s length. Higher **Overlap Ratio** results in a higher range of shared samples between each individual colliding frame.
- After creation every frame, the samples of the frame are multiplied by the *hamming window* to “soften” the transitions and avoid creation of *clipping*.

DYNAMIC SAMPLE ELIMINATION (CONTD.)

- The method has a single parameter which determines the percentage of samples that are removed from each frame. This variable is called ***Scaling Factor***, usually this value is between 0.01 (1%) to 0.05 (5%). If this factor is too high, it is going cause *clipping* in the audio due to loss of data and add noise at certain frequency bands which is very hard to remove .
- The compression works by calculating the total amount of samples that are going to be removed, which is calculated using:

$$\text{Samples to Remove} = \text{Frame Length} * \text{ScalingFactor}$$

- After which the thresholds for sample search is calculated. This is done by calculating the *Logarithmic Energy* of the frame which is calculated as

$$\text{Log Energy} = 20 * \log_{10} (\text{frameMagnitude}^2)$$

- Further, the energy is normalized to zero

$$\text{NLogE} = \text{LogEnergy} - \max(\text{LogEnergy})$$

DYNAMIC SAMPLE ELIMINATION (CONTD.)

- After this calculation mean of the $NLogE$ is found, upper and lower thresholds are determined by multiplying the $avg(NLogE)$ value by 1.19 and 0.9, respectively. These values are picked to roughly simulate the non-linear range of human sound discrimination in time.
- Any sample with energy that falls within these boundaries are marked and tested to find if the sample is within a state of transition. And once a sample has been determined to be in transition, the sample is marked for removal.
- Once enough samples have been marked for removal, these samples are removed, and neighboring samples are “softened” by a margin of the removed sample’s magnitude.
- The neighboring samples are determined by a percentage of the total length of the frame. Smoothing is done by adding a portion of the magnitude of the removed sample and decreasing this portion as the samples.
- This smoothing process heavily reduces the ripple effect that comes from annihilating seemingly random samples but does not negate it completely.

OVERLAP AND ADD

- After the frames have been reduced, the audio signal is rebuilt using the **Overlap and Add** method.
- This method reconstitutes the original signal with slight data loss at the beginning and at the end of the signal, although this loss is very minimal and near negligible in this case.

PRE-PROCESSING

```
[rec, fs] = audioread('sample.wav');  
rec = highpass(rec,100,fs); %remove DC and 60 Hz hum  
  
%Convert to mono (rastarize channels)  
channelCount = length(rec(1,:));  
if(channelCount ~=1 )  
    rec = sum(rec,2)/channelCount;  
end
```

FRAME BUILDING

- **Consider:**

`Frame Shift (Seconds) = Frame length (seconds) * OVERLAP_RATIO;`

`Frame Shift Length = frame Shift *fs;`

`Frame Shift Count = ceil(frame Shift Length);`

- After this we run a “for” loop to make multiple frames and these frames are further processed for compression.

FRAME REDUCING

- Dynamic Sample Elimination is used here to create new frames.

```
reducedFrames = []; %Process frames list
for frame = 1:length(frames(:,1))

    [reducedFrame, newFrameLength] = DynamicSampleElimination(frames(frame,:), SCALING_FACTOR);
    reducedFrames(frame,:) = reducedFrame; %Save reduced frame to list

    clc;
    f = sprintf('Reducing Frames: %d / %d', frame, length(frames(:,1)));
    disp(f);

end
```

- After this we get new sampling frequency of:

$$f_s * (1 - \text{SCALING_FACTOR})$$

FRAME PROCESSING AND REBUILDING

- Overlap Add method is used to combine all the reduced frames in one variable, i.e. Rebuilt signal.
- Further we use DCT on this rebuilt signal for Bandwidth compression and get the final output.

```
fs = ((length(rebuiltSignal)/length(rec)) * fs);  
[downsamplescompressed, compressed] = BandwidthCompression(rebuiltSignal, newframeLength, CAST_FACTOR);  
audiowrite('new3.wav',downsamplescompressed, ceil(fs/CAST_FACTOR))
```

- In Bandwidth Compression we use DCT and IDCT to down sample by ***CAST FACTOR***.

RESULTS:

- Sample Audio:



- Compressed file (1):



```
SCALING_FACTOR = 0.05; %DSE Factor  
CAST_FACTOR = 2; %DCTC Factor  
FRAME_LENGTH_SECONDS = 0.005;  
OVERLAP_RATIO = 0.1;
```

- Compressed file (2):



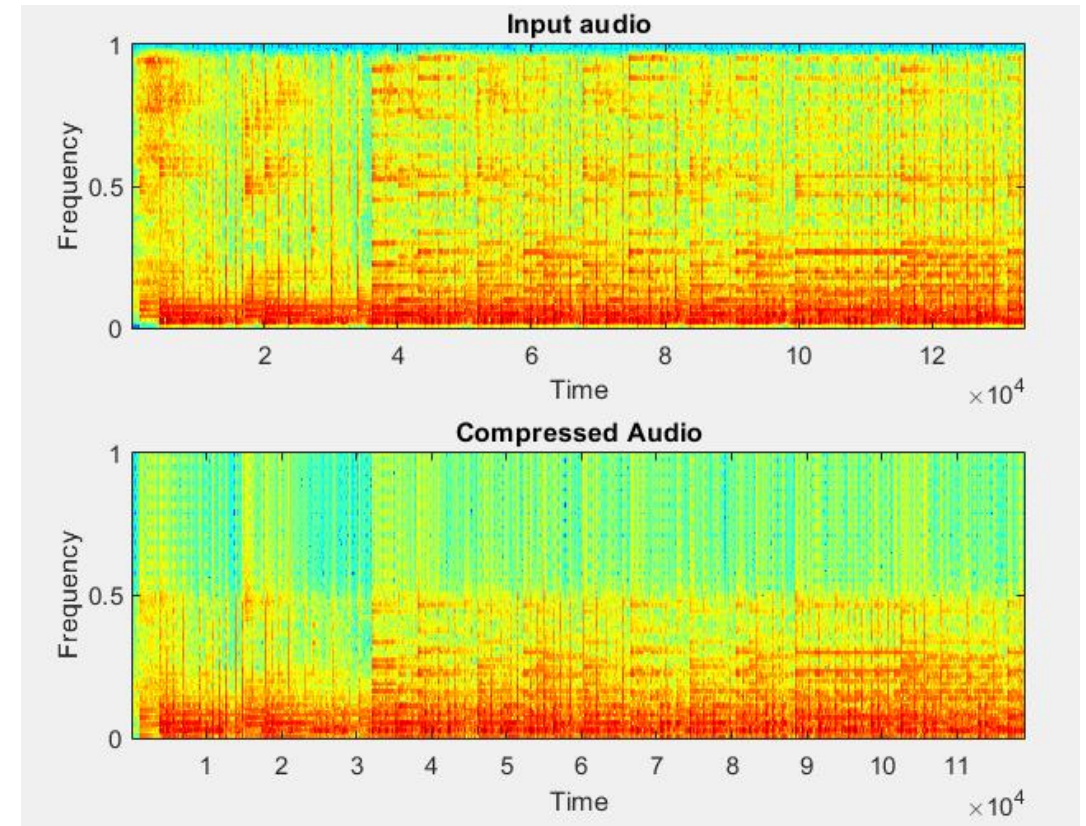
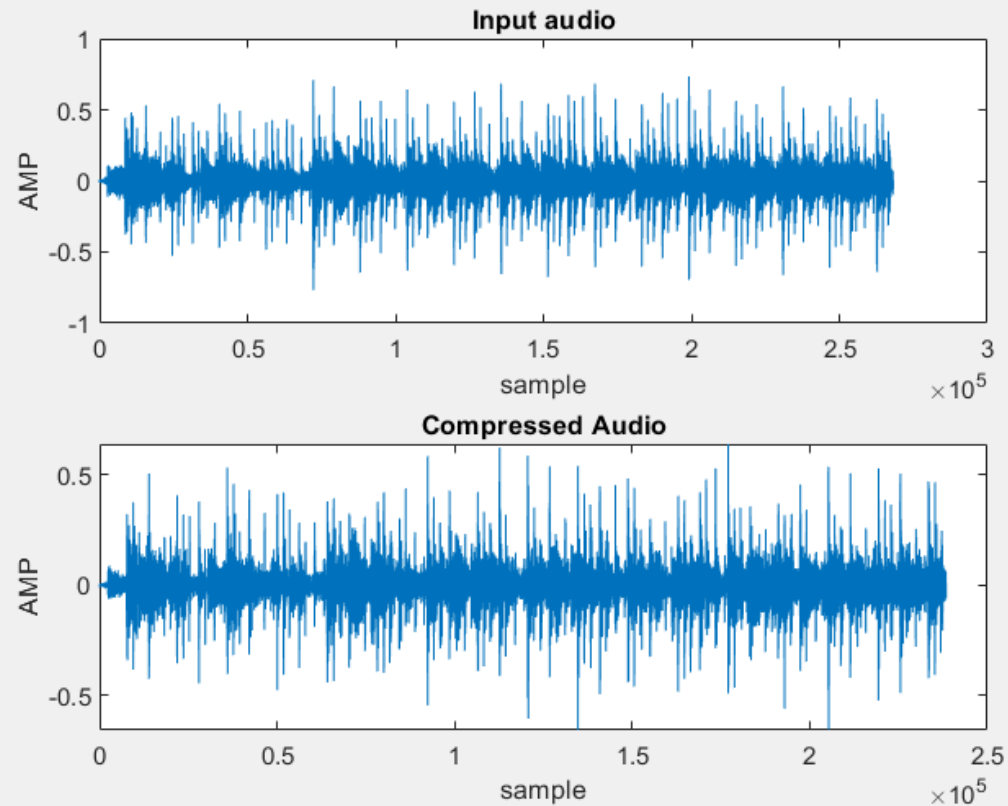
```
SCALING_FACTOR = 0.05; %DSE Factor  
CAST_FACTOR = 4; %DCTC Factor  
FRAME_LENGTH_SECONDS = 0.005;  
OVERLAP_RATIO = 0.1;
```

- Compressed file (3):

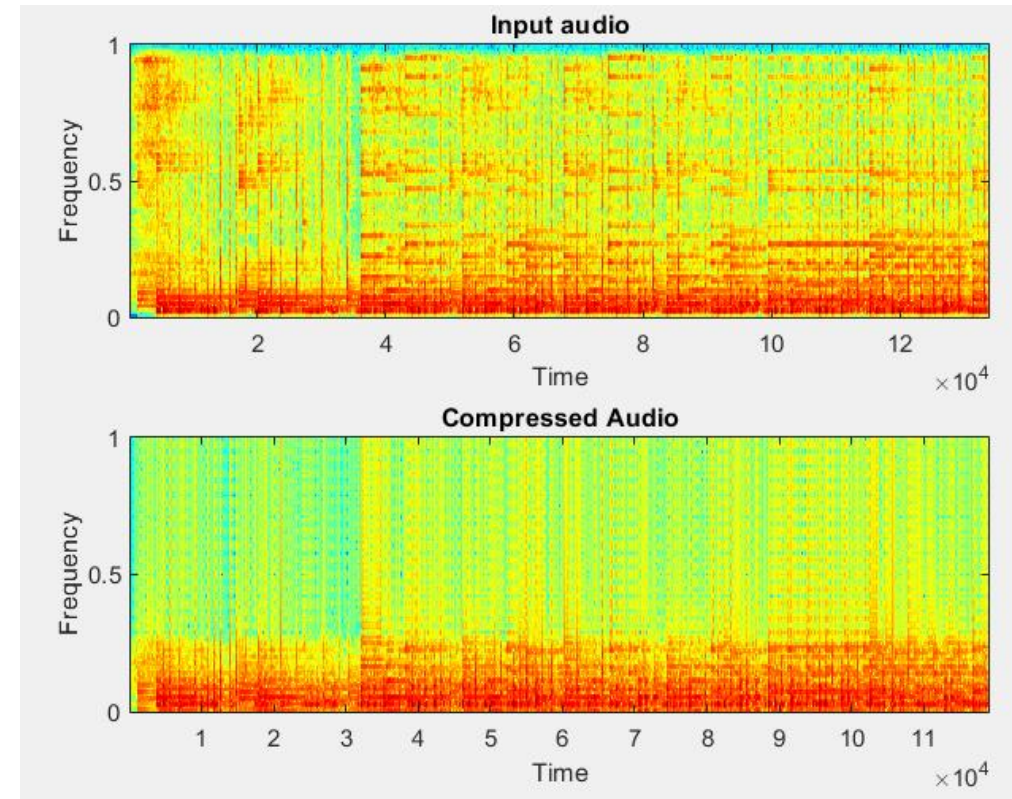
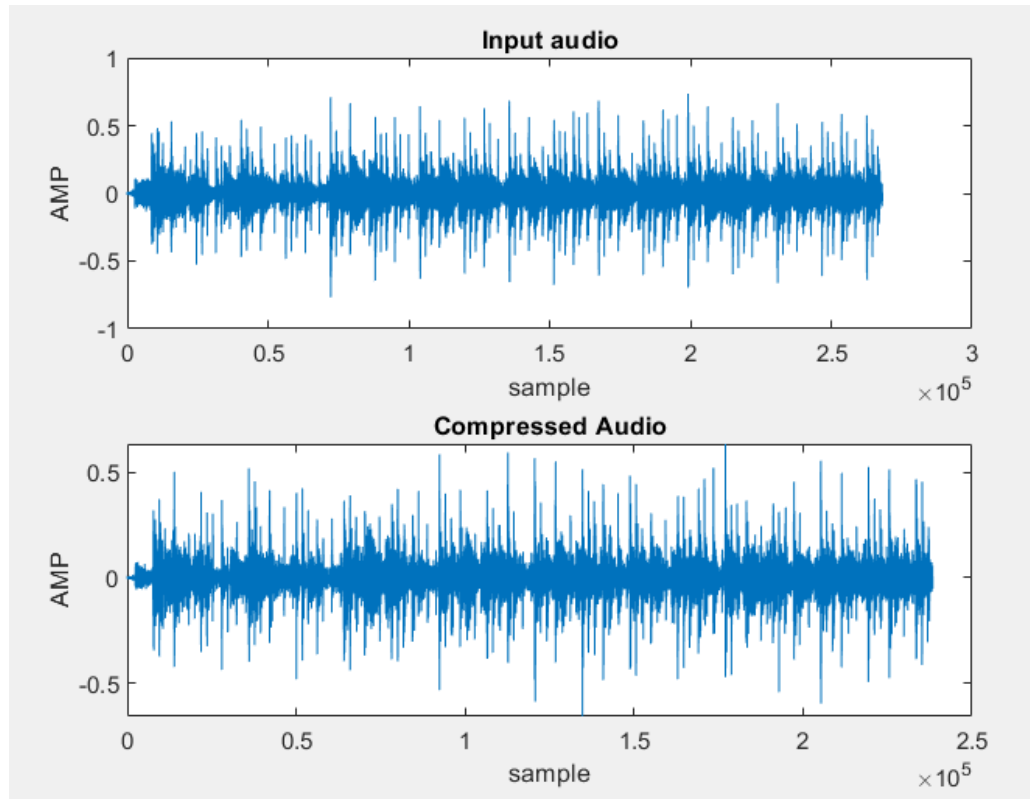


```
SCALING_FACTOR = 0.05; %DSE Factor  
CAST_FACTOR = 2; %DCTC Factor  
FRAME_LENGTH_SECONDS = 0.005;  
OVERLAP_RATIO = 0.2;
```

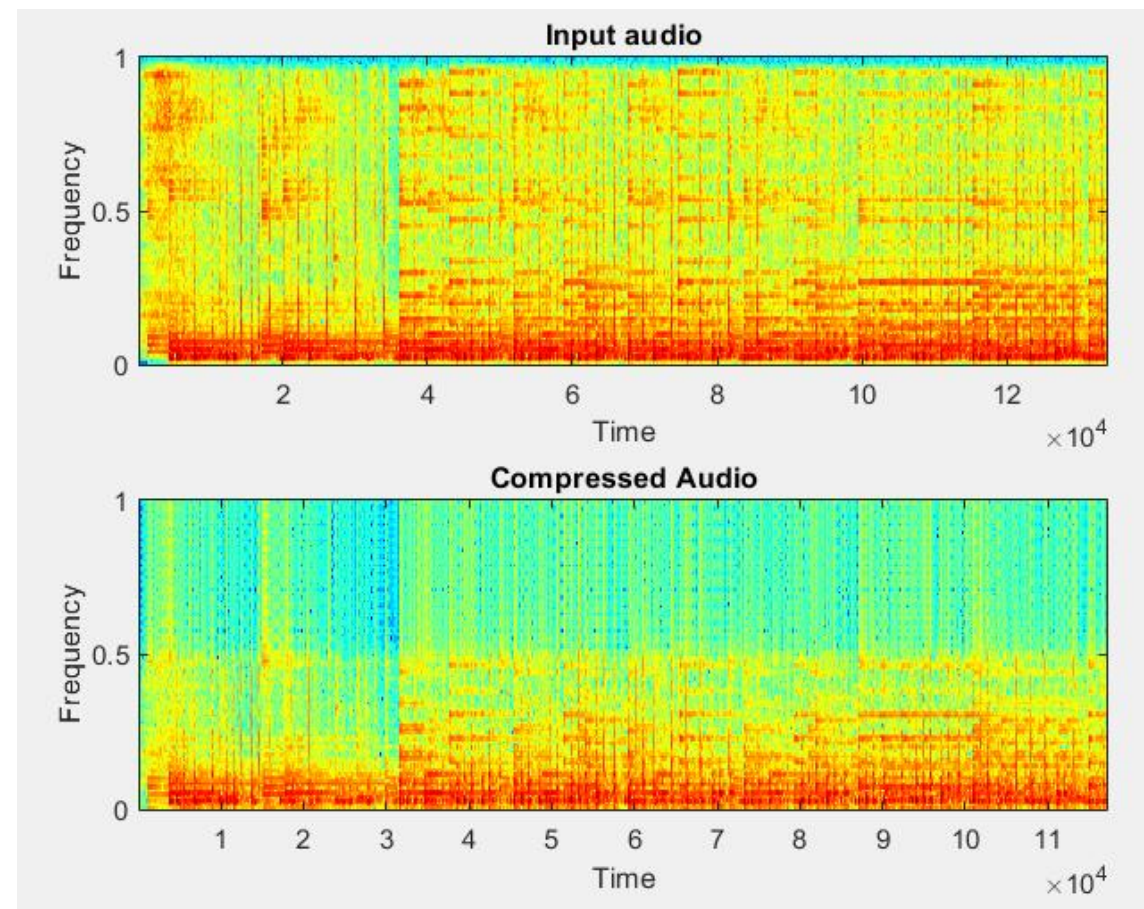
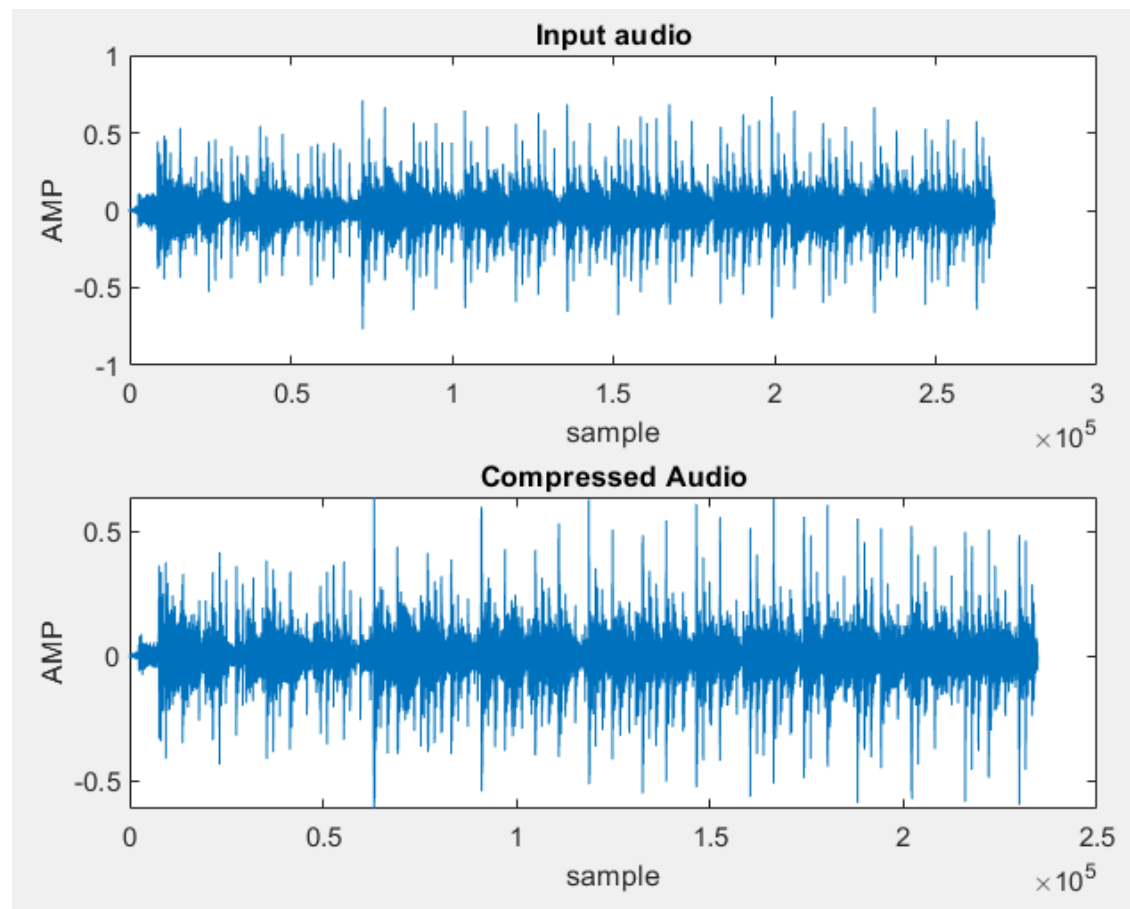
```
SCALING_FACTOR = 0.05; %DSE Factor  
CAST_FACTOR = 2;      %DCTC Factor  
FRAME_LENGTH_SECONDS = 0.005;  
OVERLAP_RATIO = 0.1;
```










```
SCALING_FACTOR = 0.05; %DSE Factor  
CAST_FACTOR = 4; %DCTC Factor  
FRAME_LENGTH_SECONDS = 0.005;  
OVERLAP_RATIO = 0.1;
```




```
SCALING_FACTOR = 0.05; %DSE Factor  
CAST_FACTOR = 2; %DCTC Factor  
FRAME_LENGTH_SECONDS = 0.005;  
OVERLAP_RATIO = 0.2;
```



FILE SIZE COMPARISON

	BandwidthCompression Type: MATLAB Code		Date modified: 02-12-2021 14:48 Size: 434 bytes
	Comp1		Length: 00:00:33 Size: 232 KB
	Comp2		Length: 00:00:33 Size: 116 KB
	Comp3		Length: 00:00:33 Size: 229 KB
	DynamicSampleElimination Type: MATLAB Code		Date modified: 02-12-2021 14:48 Size: 3.35 KB
	Main Type: MATLAB Code		Date modified: 02-12-2021 20:16 Size: 2.83 KB
	sample Kevin MacLeod	Album: YouTube Audio Library Genre: Cinematic	Length: 00:00:33 Size: 1.02 MB

SCOPE OF IMPROVEMENT

- Parameters doesn't give output file size directly.
- Rasterization of the input is done i.e.; multiple channel audio is reduced to mono-channel and the output is also mono-channel.
- Further work can also be done for compression of larger files as the model sometimes crashes for bigger files .
- There is a high scope of optimization of the presented model for faster processing.
- More improvement can be introduced in the algorithm for better output audio quality.

REFERENCES

- [1] Audio Compression, Henrik Hofling, Teodor Berglund, Axel Vaara, 2002,
<http://www.signal.uu.se/Courses/CourseDirs/SignSyst/rapporter/AudioHBV.pdf>
- [2] Digital Audio Compression, Davis Yen Pan, <http://www.dsp-book.narod.ru/DAC.PDF>
- [3] Audio Compression based on discrete cosine transform, run length and high order shift encoding,
Zainab T. DRWEESH, Loay E.GEORGE, July 2014,
http://www.ijeit.com/Vol%204/Issue%201/IJEIT1412201407_08.pdf

Thank You !