Perseverance of the Audio Data using RNN Implied Matrix Segmentation based Lossless Encoder

Asish Debnath and Uttam Kr. Mondal

1 Introduction

Lossless encoding [4] is the compression technique used when it's required to preserve the quality of the original and the decompressed material. Furthermore, it reduces the file's size without altering the data's original content. In essence, lossless audio puts preservation of detail above file size reduction [6]. Today, a variety of classical [9, 11] and neural network-based [10] techniques are utilised to handle a variety of problems. RNN is a special type of neural network which is being applied particularly to these kinds of audio encoding, privacy as well as perseverance problems. Each layer's output is recorded and sent back into the system's input via RNN [12], which uses this principle to forecast each layer's output.

The proposed lossless audio encoder is designed utilising 2D matrix segmentation based on recurrent neural networks, i.e. RNN. The encoder technique consists of the following 3 phases.

In the first stage, audio samples are divided into three components.

(a) The signed integer parts.
(b) The first two digits following the decimal point, that is, the tenths and hundredths.
(c) Next two digits, i.e., digits of the thousandths and ten thousandths positions.

Construct a 2D matrix to represent these data. This statistical technique was developed utilising a regression model built on RNN. Each of these integers is transformed into a 7-bit binary stream and represented in the latent space in the second step.

A. Debnath (✉) · U. Kr. Mondal
Vidyasagar University, Midnapore, West Bengal, India

U. Kr. Mondal
e-mail: Uttam_ku_82@yahoo.co.in

J. K. Mandal et al. (eds.), *Proceedings of International Conference on Network Security and Blockchain Technology*, Lecture Notes in Networks and Systems 738,
https://doi.org/10.1007/978-981-99-4433-0_11

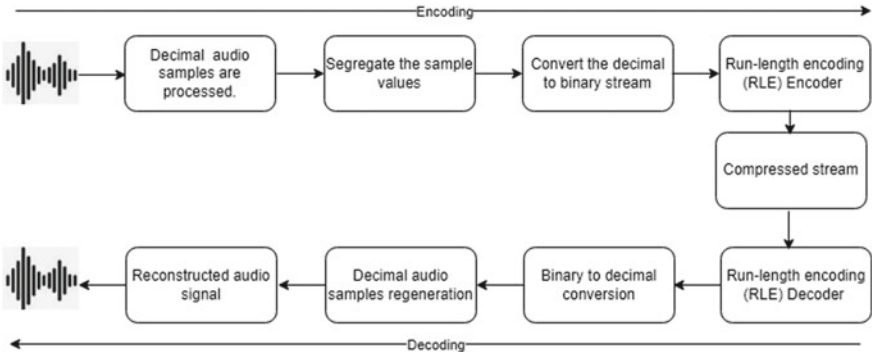

Fig. 1 Flow diagram of the proposed technique

Finally, the repetition eliminating standard Run-length encoding (RLE) [5] technique is used to further compress the binary encoded streams.

The proposed encoding and decoding flow diagram is depicted in Fig. 1.

The remaining sections of this document are structured as follows: The method is described in Sect. 2. In Sects. 2.1 and 2.2, respectively, the encoding and decoding techniques are covered. Network architecture and training are displayed in Sect. 3. Section 4 presents the findings and analyses. Conclusions are drawn in Sect. 5 that follows the references.

2 The Technique

The present method produces compressed audio using a 2D matrix segmentation algorithm which is implied by the rules of RNN. Dynamic 2D matrix-based segmentation is used to divide decimal signed audio samples during the encoding stage (considered a 4-point post fraction for the system constraint). A signed integer component and two sets of double digits after the decimal point constitute each of the three sections of the audio samples. Figure 2 shows how the encoding process works.

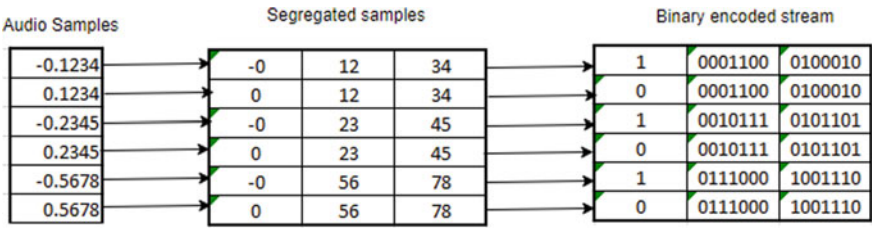

Fig. 2 Encoding example

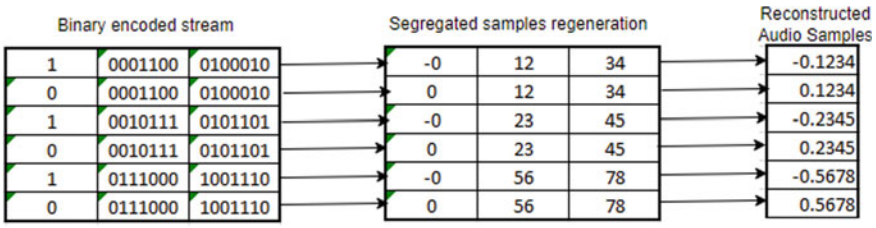

Fig. 3 Decoding example

Figure 2 explains the overview of the encoding process with 6 audio sampled values.

Encoding and decoding are done in the opposite order. The decoding processes for the previously encoded data are shown in Fig. 3.

The encoding and decoding procedures are depicted in Sects. 2.1 and 2.2 respectively.

2.1 Encoding Technique

The encoding algorithm is described in Algorithm 1. It demonstrates the segmentation of the audio samples and their eventual encoding into binary streams.

Algorithm 1: Matrix Segmentation Method
Input: A fragment of the audio

Output: Encoding audio

Method: The steps of the encoding process are as follows:

Step 1: The input audio samples are checked whether they are positive or negative. If positive go to step 1.1 (positive_module) else step 1.2 (negative_module).

The dynamic rules as listed below are used to segment each audio samples.

Step 1.1 (positive_module):

i. The whole number portion of the passed decimal number, i.e. 0 is captured. It is added to the first column of the matrix.

Let the matrix defined by $M[i][j]$, where $\boldsymbol{i}$ is the number of audio samples and j is number of columns (considering, $j = 3$). That means the matrix is 3 column matrices. Therefore, $M[i][0] = 0$.

ii. Multiply the decimal number by 10,000.
iii. Divide by 100 and take the quotient. The quotient part is moved to the second column of the same row.

$$M[i][1] = \text{Quotient part}$$

iv. Remainder part pushed to the third column of the same row.

$$M[i][2] = \text{remainder part}$$

Step 1.2 (negetive_module):
The dynamic rule listed below is used to segment each audio samples:

i. The whole number portion of the passed decimal number, i.e. -0 is captured. It is added to the first column of the matrix.

$$M[i][0] = -0$$

ii. The decimal number is multiplied by 10,000.
iii. Divide by 100 to get the quotient. The quotient part is moved to the second column of the same row.

$$M[i][1] = \text{Quotient part}$$

iv. Remainder part pushed to the third column of the same row.

$$M[i][2] = \text{remainder part}$$

Step 2: All the audio sampled values are passed to the segregation rule engine mentioned in step 1. After processing samples, the matrix formed of data passed to step 3 for binary conversion.

Step 3: The first column of first row and first column element is checked.

```
For i = 0 to length (matrix row):
    If M[i][0] < 0 then:
        Transform element at position M[i][0] with single
        binary bit '1'
    Else:
        Transform element at position M[i][0] with single
        binary bit '0'
```

Each element of M[i][1] and M[i][2] is converted to a binary stream using a 7-bit binary number.

Therefore, each audio sample is encoded as a 15-bit binary stream.

At the end of this step, a binary encoded binary stream is generated.

Step 4: Binary encoded stream is passed to run-length encoding (RLE) encoder for further compression.

2.2 Decoding Technique

Algorithm 2 describes the decompression technique. The decompression algorithm reconstructs the original audio from the encoded data stream.

Algorithm 2: Decompression technique
Input: Encoded binary stream.

Output: Reconstructed audio signal.

Method: The decoding procedure is described in following steps:

Step 1: Input encoded stream is fed to Run-length encoding (RLE) decoder. At the end of this step the binary stream is reconstructed.

Step 2: Binary stream segregated into blocks of 15 binary bits.

Step 3: Process each of the blocks using the below dynamic rule:

i. First binary bit is decoded to 0 or −0 corresponds to binary bit '0' or '1', respectively.
ii. Append decimal point after 0 or −0.
iii. Next 7 binary bits are converted to decimal values.

capture the double digit as is.

iv. Adjoining, 7 binary bits are also converted to decimal values.

capture the double digit as is.

v. Process all the blocks and regenerate the audio samples.

Therefore, finally the audio signal is reconstructed.

3 Network Architecture and Training

The sequential order of time series data, like audio, must be followed in order to be understood. Traditional feed-forward networks assume that each input is independent of the others, however in a time series setting, each input depends on the inputs that came before it.

One of the most challenging areas is deep neural network (DNN)-based sequential data processing for audio signals since DNN requires inputs with a fixed dimension. Recurrent neural networks (RNN)-based sequence to sequence learning has been proposed in machine translation to learn fixed length representations of variable length sequences [13].

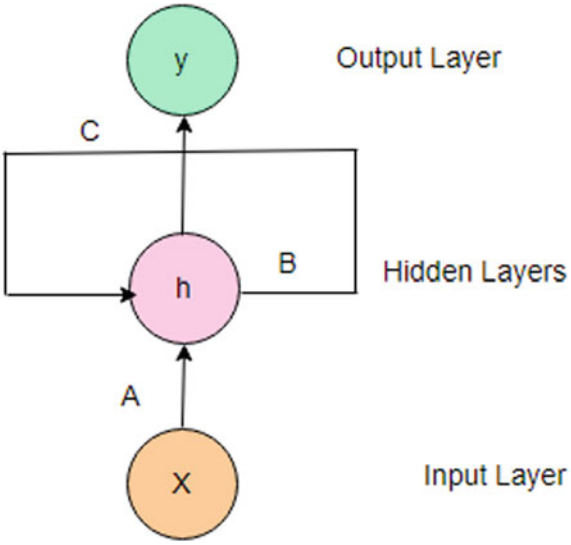

Fig. 4 Fully connected recurrent neural network

Hence, for the encoder and decoder strategy, the RNN-based network architecture is applied.

The proposed recurrent neural network model was trained with encoding and decoding algorithm described in Sects. 2.1 and 2.2 respectively. The mean squared error loss is applied to analyse the network. Single layer selected for the experiment with 16 hidden units and 3 output. All experimentation was performed with Keras using TensorFlow its back end, running under Windows 10 operating system.

A fully connected recurrent neural network is shown in Fig. 4. In this instance, "X" represents the input layer, "h" the hidden layer, and "y" the output layer. The network parameters A, B, and C are used to enhance the model's output. The input at any given time, t, is a mixture of the input at $x(t)$ and x. $(t-1)$. The output at any given time is fetched back to the network in order to improve it.

Figure 5 shows the processing in the recurrent neural network.

$$h(t) = f_c((h(t-1), x(t)) \tag{1}$$

here, $h(t)$ denotes new state, f_c denotes function with parameter c, $h(t-1)$ denotes old state, and $x(t)$ denotes input vector at time step t.

5000 epochs were considered for the experiment to evaluate the performance of the suggested model. The hyperparameters of the experiment were designed by trial and error. The configuration details of the suggested model parameters are described in Table 2. The suggested model accurately predicts and recreates the original audio data with incredibly minor deviations. The suggested model's estimated mean square error is 0.0426. We have carried out a performance evaluation of the proposed model using two additional, pre-existing prediction benchmarks, the Lasso regression, and

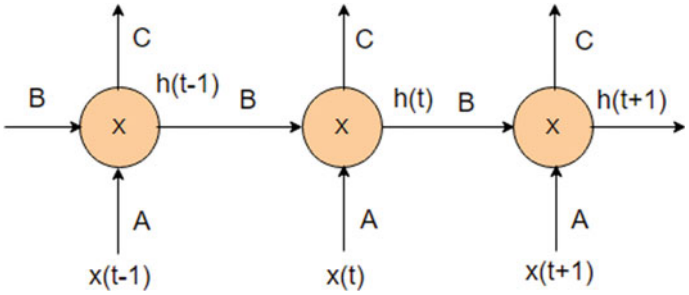

Fig. 5 Processing recurrent neural network

Table 1 Compares the suggested model's performance to those of other deep learning models

Ridge	0.0578
Lasso	0.0516
Proposed model	0.0426

Table 2 Network parameters

Learning rate	0.01
Epoch	5000
Hidden unit	16
Output	3
Timestep	1

the Ridge regression. The suggested model's loss value is contrasted with that of the two other models described in Table 1.

3.1 Dataset

For the model's training, the "audio_model_dataset" custom dataset has been created. 1000 audio files totalling 3 s each make up the dataset. There are five different categories of audio songs: (I) Rabindra Sangeet, (II) Classical, (III) Rock, (IV) Pop, and (V) Sufi. The training dataset has not divided into separate portions for testing. Instead, 25 audio songs are used, each with a standard length of 5 s, that fall into the aforementioned 5 categories.

3.2 Environment

The proposed model is programmed in Python 3.6 using the Keras and Tensorflow framework. The infrastructure used for the experiment is described below:

i. 64-bit operating system.
ii. 16 GB of RAM.
iii. Intel Core i7-4790S Processor.
iv. 1 TB Hard drive.

3.3 Network Configuration Parameters

During the experiment, various parameters have been used to train the model and measure the loss of the model and accuracy model. Present suggested model produces less loss when compared to other models of a similar nature. The list of parameters utilised in the model network is shown in Table 2.

4 Results and Analysis

Compression ratio [10, 11] is used to judge the compression capacity of a compression technique. Compression ratio is calculated as below:

$$\text{Compression ratio} = \frac{\text{Uncompressed audio file size}}{\text{Compressed audio file size}} \tag{2}$$

Therefore, after applying the compression technique, required space is to be reduced. Equation 3 represents the space saving metric respect to the compressed audio.

$$\text{Space saving}(\%) = 1 - \frac{\text{Compressed audio file size}}{\text{Unompressed audio file size}} * 100 \tag{3}$$

The compression ratio of the selected songs is compared using FLAC [3], WavPack Lossless [2], and Monkey's Audio [1]. The compression quality of the proposed method is compared graphically to that of three different systems as shown in Fig. 6. Comparing the proposed strategy to other referred techniques, Table 3 demonstrates that it has the highest compression ratio. Additionally, it offers each group the highest compression ratio (i.e. Rabindra Sangeet, Pop, Classical, Sufi, and Rock).

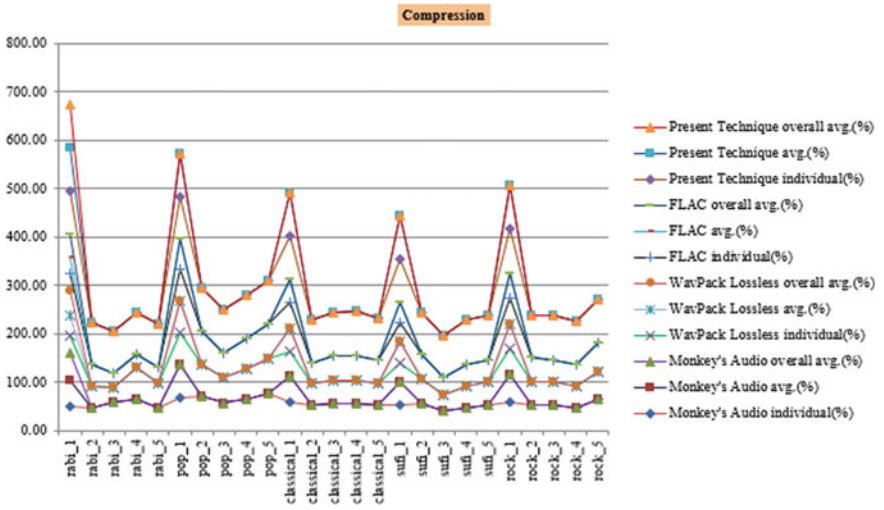

Fig. 6 Graphs showing the compression ratios for the proposed method, FLAC, WavPack lossless, and Monkey's Audio

5 Conclusion

The proposed lossless audio compression technique achieved a higher compression rate in comparison to already accessible tools for lossless audio compression. The reconstructed song is also achieved similar quality as original. The broadened focus of this research is the possibility for further compression ratio enhancement along with increasing privacy and perseverance of the audio data. In the future, it will also intend to further reduce the loss by expanding the dataset and adding more epochs.

Table 3 Compression performance

Song type	Audio files (.wav) (10 s)	Compression ratio (%)							
		Techniques							
		Monkey's Audio [1]		WavPack [2]		FLAC [3]		Present technique	
		Avg.	Over all avg.	Avg.	Over all avg.	Avg.	Over all avg.	Avg.	Over all avg.
Rabindra Tagore	rabi_1	54.89	56.35	48.46	50.14	48.18	48.28	88.02	89.81
	rabi_2								
	rabi_3								
	rabi_4								
	rabi_5								
Pop	pop_1	67.41		64.13		64.34		88.36	
	pop_2								
	pop_3								
	pop_4								
	pop_5								
Classical	classical_1	55.55		50.03		50.78		89.81	
	classical_2								
	classical_3								
	classical_4								
	classical_5								
Sufi	sufi_1	49.68		43.18		43.21		88.96	
	sufi_2								
	sufi_3								
	sufi_4								
	sufi_5								
Rock	rock_1	54.22		44.97		34.9		90.23	
	rock_2								
	rock_3								
	rock_4								
	rock_5								

References

1. https://www.monkeysaudio.com/
2. https://www.wavpack.com/
3. Coalson J (2017) Xiph.org foundation. FLAC: Free lossless audio codec. https://xiph.org/flac/index
4. Ghido F, Tabus I (2012) Sparse modeling for lossless audio compression. IEEE Trans Audio Speech Lang Process 21(1):14–28
5. Arif M, Anand RS (2012) Run length encoding for speech data compression. In: 2012 IEEE international conference on computational intelligence and computing research, pp 1–5. https://doi.org/10.1109/ICCIC.2012.6510185
6. Sharma K, Gupta K (2017) Lossless data compression techniques and their performance. In: 2017 international conference on computing, communication and automation (ICCCA). IEEE, pp 256–261
7. Nowak N, Zabierowski W (2011) Methods of sound data compression–comparison of different standards. (4)
8. Mohdar FJ, Al-Otaibi MS, Aboalsamh HA (2011) Audio compression testing tool for multimedia applications. In: Image processing and communications challenges, vol 3. Springer, pp 409–418
9. Mondal UK, Debnath A (2021) Developing a dynamic cluster quantization based lossless audio compression (DCQLAC). Multimed Tools Appl 80:8257–8280. https://doi.org/10.1007/s11042-020-09886-3
10. Mondal UKr, Debnath A, Mandal JK (2020) Intelligent computing: image processing based applications, vol 1157. ISBN 978-981-15-4287-9
11. Mondal UKr, Debnath A (2022) Multimedia Tools Appl 81(28):40385. https://doi.org/10.1007/s11042-022-12556-1
12. Sherstinsky A (2020) Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. Physica D Nonlinear Phenomena 404:132306. https://doi.org/10.1016/j.physd.2019.132306
13. Sutskever I, Vinyals O, Le QV (2014) Sequence to sequence learning with neural networks. In: Advances in neural information processing systems, vol 27
14. Pedro HTC, Larson DP, Coimbra CF (2019) A comprehensive dataset for the accelerated development and benchmarking of solar forecasting methods. J Renew Sustain Energ 11:036102