Computer Science • 26(3) 2025 https://doi.org/10.7494/csci.2025.26.3.6427

Asish Debnath
Uttam Kr. Mondal

OPTIMIZED LOSSLESS AUDIO COMPRESSION USING DCT ENERGY THRESHOLDING AND MACHINE LEARNING TECHNIQUE

Abstract *This paper proposes a novel lossless audio compression technique, utilizing the Discrete Cosine Transform (DCT) coefficient-controlled technique based on energy thresholding, an XOR-based neural network compression model, and a CNN model. Initially, the DCT is applied to the input audio signal to achieve better energy compaction, followed by transforming selected DCT coefficients into a compressed binary stream. Subsequently, this binary stream is passed to two prediction-based optimized models: an XOR model and a CNN model for further compression.The binary stream is divided into two equal pieces, the data and the key. The XOR neural network model processes the data and key to produce an compressed XORed binary stream. Using a proposed CNN architecture, this stream is further compressed with latent space representations to produce compressed audio data. The simulation findings are analyzed using various statistical and robustness measures and compared with existing approaches.*

Keywords DCT, lossless compression, audio codec, machine learning, CNN, energy thresholding

Citation Computer Science 26(3) 2025: 5–30

1. Introduction

The audio and video industries, sensor networks, the healthcare industry, and other industries are generating an increasing volume of data. Evaluating, storing, and transferring these enormous volumes of data via a network poses challenges. Large amounts of memory are required to store these uncompressed data, and considerable bandwidth is needed for data transit across a network. Compression is a size-reduction technique that is particularly useful for decreasing bit depth, dimension, and other elements of audio and image files. When the size of the compressed file is reduced, lossless audio compression ensures that the audio quality is preserved. To return the compressed file to its original state, it must be uncompressed. By eliminating less important audio data, lossy audio compression increases the compression ratio at the expense of audio quality [7]. Some data is permanently lost due to lossy compression, making it impossible to recreate the original data. Therefore, lossy compression cannot be applied when no data loss is acceptable. Applications based on lossless audio compression have become increasingly popular in recent years, with demand rising across a range of industries including defense, forensics, and medicine [26]. Furthermore, lossless audio compression is essential for numerous other applications, including recording, editing, and lossless music transmission [28]. The primary benefit of lossless compression is that data quality remains unaffected because no data is lost.

Numerous studies on lossless audio compression have been conducted up to this point. Many notable approaches related to lossless audio compression have been introduced, including IEEE 1857.2 [13], MPEG-4 ALS (LPC) [32], Enhanced SLS, and others. In addition to these techniques, several popular and contemporary audio codecs are currently available, such as Monkey"s Audio [27], Wavpack [37], and FLAC [5]. Furthermore, deep neural network technology has recently been applied to the development of lossless audio codecs. It has been found that Wavpack produces an average compression ratio above 50% [15], FLAC achieves an average compression ratio around 70% [29], and Monkey"s Audio achieves an average compression ratio of roughly 60% [29]. In 2018 [18], a neural network-based model was introduced that uses CNN to process features created from raw audio input. This technique introduces an end-to-end compression pipeline based on neural networks. However, employing this strategy does not improve the compression ratio. In 2019, a lossless audio compression approach based on Golomb codes was introduced. This paradigm uses a context dependent arithmetic encoder for compression, reducing processing complexity without appreciably improving compression. A lossless audio encoder based on dynamic cluster quantization was released in 2020 [23]. In this suggested clustering-based technique, the quantization level is set up by applying bit selection and dynamic cluster selection simultaneously. In 2021, a novel machine learning-based lossless encoder model was introduced [33]. This method uses a deep learning methodology to describe the latent space in binary form, combining recurrent neural networks (RNNs) and variational autoencoders (VAEs). However, while the compression ratio improves with the use of a deep learning approach, the model"s complexity

increases concurrently. In fact, the reconstructed audio signal may be noisier than the raw audio signal, indicating a decline in audio quality. In 2022, a lossless audio codec based on optimized graph traversal was introduced [24]. Additionally, this codec has a higher compression rate than earlier methods, utilizing a traveling approach based on graphs. A machine learning-based toolset for unsupervised learning from audio data was recently presented [9]. This technique is based on repeating sequential autoencoder technology, which learns from time-series type data by utilizing temporal motion. However, the compression ratio does not improve with this model either. Another method of audio compression introduced in 2022 is the linear predictive neural network-based encoder (LINNE) [22]. The primary predictor for this algorithm is NN processing, achieving approximately 60% audio compression. Although LINNE performs better in compression, there is no improvement in encoding or decoding times. Lossless audio codecs like Monkey"s Audio [27], TTA [35], and WavPack [37] employ adaptive filters as their predictive model, while FLAC [5], ALAC [1], and MPEG4-ALS [32] use linear predictive coding (LPC). As the residual in an LPC calculation is typically assumed to follow a Gaussian distribution, MPEG4-ALS [32] and FLAC [5] are also based on this assumption. Adaptive algorithms are utilized by Monkey"s Audio [27], TTA [35], and Wavpack [37], which are based on the Laplacian distribution. For an adaptive method, the sign algorithm (SA) [8] is a reasonable option if the residual has a Laplace distribution; nevertheless, the SA converges far more slowly than the least mean square (LMS) algorithm [11]. The IEEE 1857.2 standard [13] is another recently introduced audio codec. Applying entropy encoding results in a larger compression ratio which is roughly more than 50% [15]. However, this encoder has a lengthy encoding time because, instead of direct encoding, a preprocessing step is added to lower the dynamic range [10]. The Sparse Linear Predictor [12] audio codec boasts an enhanced average compression ratio of over 50%, but its decoding speed is average. More recently, a novel lossless audio codec based on CNN, arithmetic encoding, and the weighted tree method [6] was introduced, improving audio compression. A lossless audio codec is introduced based on two adaptive algorithms: the natural gradient sign algorithm (NGSA) and normalized NGSA. In this work, the use of a natural gradient enhances the convergence performance of the sign algorithm (SA). These techniques, which assume a p-th order autoregressive model for the input data, use multiply-add operations to compute the natural gradient at each step, greatly increasing decoding speed. However, the compression performance attained with this technique is approximately 60%. The audio codec proposed by the authors is known as the natural gradient autoregressive unlossy audio compressor, or NARU [21]. The Golomb Rice code [14] is commonly used in entropy coding because it performs best when the residual has a Laplace distribution. However, an LPC residual assumption is incorrect. To address this issue, Kameoka et al. [17] created an LPC under a Laplace distribution, which increased the compression rate.

The primary challenge lies in achieving a compression ratio greater than that of lossy compression. Currently, there is no lossless audio encoding method that can compress audio files in a manner comparable to lossy methods like MP3 and others.

It is evident from previous research that a lossless audio compression method with a high compression ratio is still needed. In this approach, traditional methods are amalgamated with machine learning techniques, particularly utilizing a Convolutional Neural Network (CNN) model, to generate superior compressed audio, which constitutes a unique compression scheme compared to existing alternatives. This study endeavours to enhance the compression ratio of the proposed lossless audio compression technique compared to state-of-the-art methods, while preserving audio quality.

In order to achieve the delicate balance between audio quality and file size, audio compression is a complex field that integrates many technical components. Audio compression's primary goals are to reduce duplication and effectively convey audio data. Spatial redundancy reduction is employed in the suggested compression method. The goal of spatial redundancy in this case is to remove redundant data using the discrete cosine transform (DCT) technique. For additional compression, neural network-based methods have also been used. While compression is useful in the music industry, misuse of it can lead to unfavourable outcomes. Excessive application of compression could inadvertently diminish the natural dynamics of the audio stream. Overuse of compression can result in a robotic, unnatural tone in audio. File size and audio quality are ultimately correlated. Higher compression ratios usually come at the expense of some quality. A trade-off between the two must be made in light of the intended usage and the resources at hand in order to achieve the ideal balance. 89.25% compression was achieved in the proposed audio codec without sacrificing audio quality.

The proposed lossless audio CODEC architecture (see Fig. 1) is designed in three steps to achieve the following objectives: i) to improve the compression ratio, ii) to maintain audio quality by minimizing loss during the reconstruction of the audio signal, and iii) to optimize the models for better signal prediction.

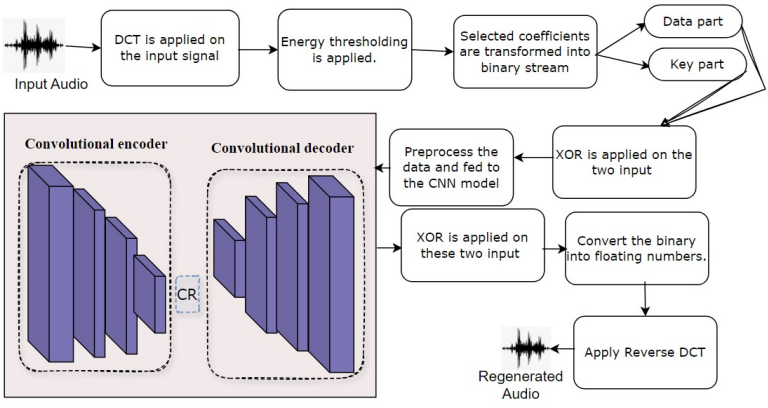

Figure 1. Proposed architecture

The key aspect that ensures losslessness in this encoding scheme is the application of an energy threshold. Lossless audio compression using Discrete Cosine Transform (DCT) coefficient energy thresholding is a technique that involves transforming audio signals into the frequency domain using the DCT, selectively retaining only significant coefficients, and discarding insignificant ones based on their energy levels. Coefficients below this threshold are deemed insignificant and are either quantized or discarded. However, the thresholding process ensures that only negligible or redundant information is removed, preserving the essential components required for faithful reconstruction of the original audio signal.

In the first stage of the proposed audio encoding technique, Discrete Cosine Transformation (DCT) [4] is applied to the audio input. DCT basis vectors [14] are generated by decomposing the signal, with the decomposition having the same number of terms as there are audio samples in the input audio signal. The coefficients in the resulting vector indicate the amount of energy stored in each component. Energy thresholding is implemented in the proposed technique. For this experiment, the number of DCT coefficients [30] corresponding to 99.99% energy of the input audio signal is determined as the threshold value. However, the energy level section cap could vary depending on demand. Selecting higher energy levels makes it suitable to reconstruct the audio signal properly without compromising quality. Consequently, the recently produced set of DCT coefficients, which preserve 99.99% of the energy present in the input signal, is good for reconstruction. During the reconstruction process, these coefficients accurately recreate the audio stream without sacrificing audio quality. The remaining 0.01% of energy is represented by the DCT coefficients, which are set to zero. The reconstructed audio quality remains unaffected by this negligible data change since it is beyond the human perception threshold level. Real-valued DCT coefficients are then converted into a binary stream using the weighted tree binary transformation [6] method. The entire created binary stream is divided into two sections: the data portion and the encryption key portion. Both parts are passed to the proposed neural network model for XOR operation [19]. The key part is stored for performing the inverse XOR operation during decompression. The XOR data is segregated and transformed into each of the 20x20 images to train and test the proposed CNN model [16] for dimensionality reduction and compressed data generation.

With a compression ratio of 89.25%, the proposed codec is closer to lossy audio codecs like Mp3, which have a ratio of above 90%. Furthermore, compared to the currently in use lossless audio codecs, its compression ratio is higher.

1.1. Notations and symbols

Table 1 provides the terms that are pertinent to this paper's abbreviations and their full definitions.

Table 1
Symbol and abbreviation form of some terms

Symbol/Abbreviation	Meaning/Full form
DCT	Discrete cosine transformation
CNN	Convolutional Neural Network
CR	Compressed representation
DNN	Deep neural networks
RELU	Rectified linear unit
PNSR	Peak signal-to-noise ratio
NCC	Normalised cross correlation
MAE	Mean absolute error

2. The technique

The proposed technique is formed with 3 stages. Discrete cosine transformation is applied on the input audio and vector generated with the DCT [39] coefficients. The coefficients in the resultant vector show how much energy is stored in each component. The energy thresholding technique is applied to control the compression level along with maintaining quality. In the next step, selected DCT coefficients are converted into binary stream using the weighted tree binary transformation technique [6]. The weighted tree binary transformation algorithm transforms DCT coefficients into a binary stream. It has been observed that the range of the generated DCT coefficients for the audio dataset utilized in the simulation is -1 to $+1$. The weighted tree technique of creating tree nodes entails splitting the 0 to 1 range into two parts and computing the mean, which serves as a key. These two means were subdivided into four, eight, and sixteen parts, respectively. The binary bits 0 and 1 are utilized to represent the weight of the key's left and right edges, or mean, respectively. Similar calculation is applied for -1 to 0. Table 2 displays the node and associated weighted path binary streams in Section 2 (The technique) of the paper. Table 2 shows the mapping for the described approach between the binary path values and the corresponding tree node or key. The weighted binary coding tree is depicted in Figure 2.

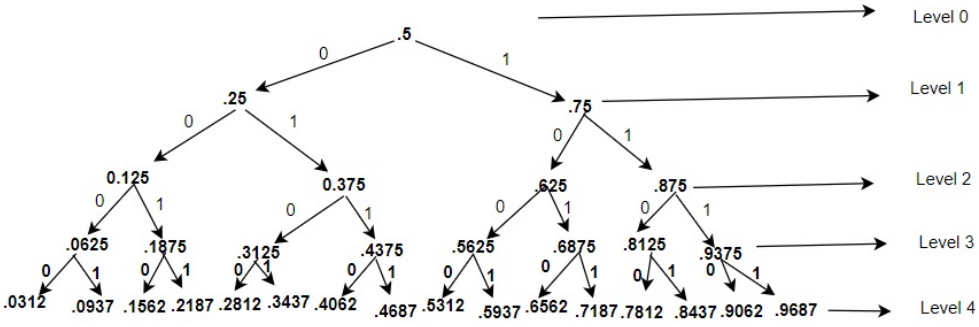

Figure 2. Weighted tree encoding

Table 2
Binary mapping of cluster midpoint

Cluster midpoint value	Binary coded form
0.0312	0000
0.0937	0001
0.1562	0010
0.2187	0011
0.2812	0100
0.3437	0101
0.4062	0110
0.4687	0111
0.5312	1000
0.5937	1001
0.6562	1010
0.7187	1011
0.7812	1100
0.8437	1101
0.9062	1110
0.9687	1111

The binary stream as a whole is split into two parts: the encryption key part and the data part. Therefore, the data component and the encryption key portion comprise the two halves of the generated binary stream. Then, pass the data part stream and encryption key stream to the proposed XOR model. Keep the key part for future XOR decomposition model. The XOR model output segregated into 20×20 rows. Each of these rows converted into 20×20 images. These images are to train the CNN model for dimensionality reduction.

2.1. Discrete cosine transform

The prevalent form utilized in this research paper is 1D DCT type II [4]. Let:

$$y \in H^{pxq}$$

is a matrix. It is made up of entries $y_{m,n}$ where $m = 0, ..., p-1, n = 0, ..., q-1$. A matrix is produced by using the Discrete Cosine Transform type II (DCT II) on y.

$$Y \in H^{pxq}$$

DCT coefficients made up the matrix. Matrix $Y_{c,d}$, where $c = 0, ..., p-1, d = 0, ..., q-1$ is formed by the Equation (1):

$$Y_{c,d} = \sum_{m=0}^{q-1} \sum_{n=0}^{p-1} y_{m,n} cos \left[\frac{\pi}{q} \left(n + \frac{1}{2} \right) d \right] X cos \left[\frac{\pi}{p} \left(m + \frac{1}{2} \right) c \right] \tag{1}$$

With IDCT-II (Inverse Discrete Cosine Transform type II), the original data y can be recovered given DCT coefficients [34] Y in the manner described below:

$$y_{m,n} = \sum_{c=0}^{p-1} \sum_{d=0}^{q-1} Y_{c,d} cos \left[\frac{\pi}{q} \left(n + \frac{1}{2} \right) d \right] X cos \left[\frac{\pi}{p} \left(m + \frac{1}{2} \right) c \right] \tag{2}$$

As a result, the formulas for IDCT-II and DCT-II are rather similar. They hold true for matrices. It is simple to obtain the vector DCT-II and IDCT-II formulae by setting $q = 1$. The preceding formulas are analogous to those for DCT-II and IDCT-II for tensors of higher order. The Binary Transformation Module receives these DCT coefficients.

These binary stream sent to the XOR model (Section 2.2) for compression.

2.2. XOR compression model

XOR model implemented by multi-layer perceptron (MLP) i.e., a deep learning model. The XOR model is a DNN model that takes two binary inputs and returns a binary output. It returns 1 if exactly one of the inputs is 1, and 0 otherwise. An artificial neuron type utilized in machine learning is called a perceptron. It generates an output by taking a number of inputs, summing them up, applying weights to them, and then running them through an activation function. A perceptron can be represented mathematically as:

$$output = activation function(weights \cdot inputs + bias) \tag{3}$$

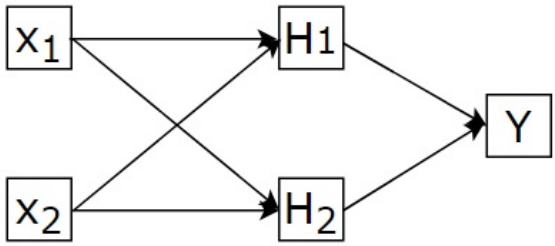

Figure 3. Multi-layer perceptron structure

The input layer is the top layer in this structure (Fig. 3). The output layer is the third layer, and the hidden layer is the second layer. Adam serves as the optimizer, while MSE is taken into consideration as the loss function. Present model utilizes 200 epochs. ReLU serves as the activation function in both the input and hidden layers. In the output layer, the sigmoid is employed as the activation function. Table 3 shows all the details of the XOR model configuration parameters.

Table 3
XOR Model configuration parameters

Parameter name	Value
Inner layer activation function	ReLU
Final layer activation function	Sigmoid
Epoch	200
Loss function	MSE
No of hidden layer	2

2.3. CNN encoding decoding model

The CNN encoder that is being proposed consists of four convolutional layers and steps. Activation layers, four deconvolution layers, and up-sampling layers make up the decoder network. Selecting the most informative characteristics while eliminating any that are unnecessary is the aim of feature selection. Features were extracted from the input signals using four two-dimensional convolutional layers. The signal reconstructor for the CNN compression decompression model was the deconvolution version of the CNN feature extractor. The CNN encoder decoder consisted of four convolutional layers, with an input dimension of 20×20, and a dimension of 2×2 for the latent space variable Z. The model's Adam optimizer used learning rate rescheduling for the purpose of fine-tuning. As the CNN model advances through the network to the final layers, it starts its training process with the preprocessed file input in the (.csv) format, where each row has 400 columns carrying pixel values and is regarded as a $20X20X1$ picture. Feature maps are generated by convolution using a predefined set of filter banks. Rectified linear unitr activation function (RELU) is then used to carry out activations. Next, a stride-rate of 2 and a stride window size of (2,2) are used for the pooling procedure. As a result, the subsequent data are subsampled twice. A two-dimensional latent space is created by compressing a 400-dimensional input vector. Then, the decoder network (s) upsamples the input feature map. Following this phase, which first creates sparse feature maps, a trainable decoder filter bank is used to create dense feature maps. A trainable sigmoid function receives the highly detailed representation of the high-dimensional feature map(s).

2.4. Encoding and decoding algorithm

The encoding and decoding algorithm is described in Algorithm 1.

Algorithm 1 Encoding decoding algorithm

Input: *A strip of audio signal*
Output: *Regenerated audio signal*
Method: *The encoding procedure is given in following steps:*

1. DCT transformation 1D Type II DCT [14] transformation is applied on the input audio. Let, number of audio samples range denoted by $x(p)$ where $p = 0, 1, ..., N-1$. DCT of the data sequence is denoted by $X(q)$ where $q = 0, 1, ..N-1$.

The DCT transformation is described by below Equation (4):

$$X(q) = e(q) \sum_{p=0}^{N-1} x(p) cos \left[\frac{(2p+1)\pi q}{2N} \right], q = 0, 1, .., N-1 \tag{4}$$

where

$$e(q) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if q=0} \\ 1 & \text{otherwise} \end{cases} \tag{5}$$

A matrix of DCT coefficients generated which have different energy levels.

2. DCT coefficients selection Using energy level thersholding, required number of DCT coefficients generated which is enough to regenerate audio by applying inverse DCT on these coefficients. Here, we have selected a list of DCT coefficients which represents 99.99% of the energy in a sequence.

3. Transform DCT coefficients into binary Weighted tree binary conversion [4] method applied to transform the DCT coefficients into binary stream.

4. XOR encoder model The above generated binary stream segregated into 2 part: one part denoted as data part and the other part is key part. key part is stored for inverse the XOR operation in decoder model. These two stream is fed to the XOR model for generating the encoded compressed stream. Proposed XOR model structure is described in section 3.2.

5. CNN encoding decoding model XOR encoded binary stream is preprocessed and segregated to 20X 20 in each row. Then each of the row is tranformed into images. These images are used for train and test the CNN model. CNN encoder part generates the dimensionality reduced data. CNN decoder the upscale and regenarate the data with original dimension. CNN encoder decoder model is described in Section 3.3.

6. XOR decoder model Binary stream generated from CNN model is passed to the XOR decoder model. Also, the key part is sent to the model. Then, reverse XOR operation is perfomed and binary stream generated.

7. Inverse binary operation Above generated binary stream and the key stream is combined to generate the original binary stream generated by step 2. Transform the binary stream into decimal of of the DCT coefficients using the weighted tree method.

8. Inverse DCT transformation The inverse DCT is applied to the retained compressed coefficients, resulting in an exact replica of the original audio signal. Since no information is lost during encoding and decoding, the process is considered lossless.The DCT inverse transformation is described by Equation (6):

$$x(p) = \frac{2}{N} \sum_{q=0}^{N-1} e(q) X(q) cos \left[\frac{(2p+1)\pi k}{2N} \right], p = 0, 1, .., N-1 \tag{6}$$

where

$$e(q) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if q=0} \\ 1 & \text{otherwise} \end{cases} \tag{7}$$

Therefore, the input audio is reconstructed.

3. Experimental setup

The following section covers a number of necessary components for the experiment, such as the environment setup, the dataset, the data preprocessing, the tools and software used, etc.

3.1. Environment

The system used for the experimental procedure has the hardware and software setup information listed below.

3.1.1. Hardware setup

The hardware configuration used for the simulation is covered in this section.

- 11th Gen Intel(R) Core(TM) i5-1135G7. The processor has a clock speed of 2.42GHz.
- 1 TB hard drive serving as secondary memory.
- 8 GB of RAM serving as primary memory.

3.1.2. Software setup

This section describes the several softwares that are utilized in the simulation process.

- 64-bit operating system.
- Tensorflow/Keras [36](version 2.12.0) and Python 3.6 for implementing the audio codec. TensorFlow platform's high-level API is called Keras. We have employed it because it offers a user-friendly, extremely effective interface for resolving machine learning (ML) issues, with a particular emphasis on cutting-edge deep learning. Keras carried out every stage of the suggested model's implementation, including data processing, hyperparameter tuning, and deployment.
- Audacity software version 3.6.1 [3] for recording the songs in the audio dataset.Songs are recorded using Audacity (Audacity 3.4.2) in .wav format.

3.2. Datasets

The CNN encoder decoder model was trained using 500 audio song recordings with a duration of 0.02s to 0.03s for batch processing and consistent processing from the 44.1kHz sampled DCTXORCNN_audio_training_dataset. There was no splitting of the training dataset into a testing data set. Rather, an alternative testing dataset was used to assess the model's efficacy. A testing dataset called DCTXORCNN_audio_-testing_dataset was produced using the same recording parameters as the training dataset. It has 25 audio strips from the Rock, Classical, Rabi, Pop, and Ghazal categories. Both sets of audio files contain a 44.1 kHz sampling rate and a ".wav" file extension. The signals are converted from stereo to mono.

3.3. Data preprocessing

The binary stream generated by XOR compression model is preprocessed and divided into several rows, each of which contains 400 columns, similar like a $20 \times 20 \times 1$ b/w image. These images are used as input to the CNN based encoder. The image fits well into the CNN framework for training. In order for our binary audio data stream to fit inside CNN, it needs to be represented as the same format as picture. The binary input stream from the tree-based encoding part is preprocessed before sending to the CNN model framework. Total bit stream is divided into even size blocks of 400 bits. Each bit block represents the intensity value like a pixel in an image. Therefore, each block represents a black and white picture of size $20 \times 20 \times 1$. These intermediate binary preprocessed datasets corresponds to DCTXORCNN_audio_training_dataset and DCTXORCNN_audio_testing_dataset stored as .csv file and named as CNNENCDEC_training_dataset and CNNENCDEC_testing_dataset respectively.

3.4. Training

Tensorflow/Keras is the framework used to build and train the whole CNN network [36]. Tensorflow/Keras substantially facilitates the ability to design network layers and train the network in accordance with the recommended parameters. Consequently, training occurs after convergence, and a noticeable reduction in training loss is observed. Evaluating, reviewing, and contrasting the overall outcomes with the predetermined benchmark results is the last step.

3.5. Details of recording parameters

The DCTXORCNN_audio_training_dataset and DCTXORCNN_audio_testing_dataset preparation processes are covered in detail in this section. Using the Audacity program (version 3.6.1), computer-played music is recorded as a.wav file. Using Audacity, music is recorded in.wav format, resulting in a 44100 Hz sample rate data collection. The wav file format is used for recording the audio tracks used to prepare the dataset. The audio recordings are stored in the format without any bitstream compression. As a result, there is no compression present in the audio dataset. The number of audio samples recorded in one second is referred to as the "sample rate". On the other hand, a sample is a snapshot of the audio that is taken at a specific time; the sample rate controls how frequently these snapshots are taken. The higher the sample rate, the more accurate the audio representation. For all song categories, a uniform sampling rate of 44100 (44.1 kbps) was used to build the audio dataset. This means that 44,100 audio samples were taken every second. The training dataset consists of audio songs that last approximately two seconds, whereas the testing dataset lasts approximately ten seconds. There is only one (1) mono audio channel on every recorded audio track. Bit depth is another important consideration in digital audio. It discusses the number of bits that are utilized to encode every audio sample. A 16-bit bit depth was used in the generation of the suggested audio dataset. The audio dataset we used contains five different music categories: pop, ghazal, rock, classical,

and Rabi. The bit rate, another crucial aspect of audio, is the quantity of bits needed to describe one second of audio. It is calculated as the bit depth multiplied by the sample rate. A 16-bit audio recording at a 44.1 kHz sample rate in our audio dataset yields a bit rate of 705 kbps. The Table 4 displays the configuration values for these parameters.

Table 4
Dataset recording parameters

Audio recording parameter	Values
Recording format	.wav
Sampling rate	44100 Hz
Training dataset recording time	2 seconds
Testing dataset recording time	10 seconds
No. of channel	Mono (1)
Bit depth	16 bits
Categories of song	5
Bit rate	705 kbps

4. Results and analysis

The DCT encoder receives each audio file from the DCTXORCNN_audio_training_-dataset. Binary patterns are used to encode the corresponding list of DCT coefficients. Each binary data stream is now divided into two pieces, which are then utilized as the XOR model's training dataset. Next, every binary data stream that has been subjected to XOR operations matches every audio file in the CNNENCDEC training dataset, which is referred to as CNNENCDEC _training_dataset. Thirty percent is used for validation and seventy percent is used for training the CNNENCDEC model. The DCTXORCNN_audio_testing_dataset is used to prepare the CNNENCDEC _testing_dataset, which is the testing dataset for the CNNENCDEC model. The CNNENCDEC model is tested independently using this testing dataset. Three stages of the suggested codec are assessed throughout the experiment in order to compute the compression independently. The results are explained in detail in Section 5.1.

4.1. Experimental results

The effectiveness and stability of the proposed model which combines the CNN, XOR, and DCT compression models are assessed using a number of parameters. Three well known lossless audio compression methods like Monkey"s audio, Wavpack Lossless, and FLAC are taken into consideration as reference systems in order to evaluate the efficacy and capacity of the suggested model. Furthermore, Daniela N. Rim et al.'s [33] deep neural based audio compression model was used to assess the effectiveness and reconstruction capabilities of the suggested model [6]. Throughout the experiment, DCTXORCNN_audio_training_dataset and DCTXORCNN_-audio_testing_dataset were used continuously.

The overall average compression evaluated for the current proposed architecture with three referred systems is displayed in Table 5. The experiment result set makes it evident that the current model achieves an average compression rate of 89.25%, which is higher than the rates achieved by Wavpack Lossless, FLAC, and Monkey's Audio over the 25 songs in the five categories that make up the testing dataset, which are, respectively, 54.89%, and 72.01%, and 57.82%.

Table 5
Average compression comparison with respect to referenced systems

Metrics	Monkey's Audio [27]	WavPack [37]	FLAC [5]	Proposed model
Compression (%)(average)	57.82	54.89	72.01	89.25

Table 6 displays the category-wise average compression evaluated for the current framework with three referred systems. It is clear from the experiment result set displayed in Table 6 that the present model delivers more compression in every testing dataset category.

Table 6
Groupwise compression of the proposed model with respect to referenced systems

Method	Rock	Classical	Rabi	Pop	Ghazal
Monkey's Audio [27]	57.43	57.45	59.19	57.40	57.67
WavPack [37]	54.06	55.34	55.28	54.25	55.56
FLAC [5]	72.76	71.46	71.98	72.09	71.78
Proposed model	89.04	88.41	90.01	89.67	89.13

Figure 4 shows the categorywise average compression for the proposed technique.

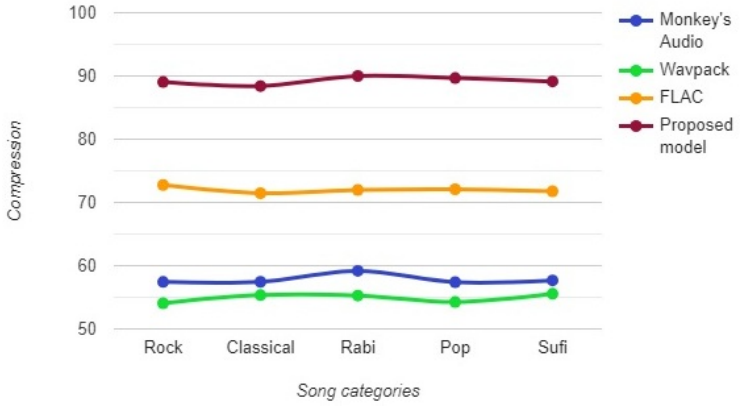

Figure 4. A graphic illustration of the suggested technique's categorywise average compression

The proposed method's average PSNR [20] value of 56.19 db is greater than that of Wavpack Lossless, FLAC, and Monkey's Audio, which are, respectively, 54.07 db, 52.10 db, and 53.14 db. Table 7 presents the experimental PSNR values for the proposed and current methods.

Table 7
PSNR comparison with respect to referenced systems.

Metrics	Monkey's Audio [27]	WavPack [37]	FLAC [5]	Proposed model
PSNR (db) (average)	54.07	52.10	53.14	56.19

The Table 8 displays the category-wise average PSNR calculated for the proposed framework with three referenced systems. It is quite clear from the experiment result set displayed in Table 8 that the present model yields higher PSNR in each category. The categorywise average PSNR comparison is shown in Figure 5.

Table 8
Groupwise PSNR comparison of the proposed model with respect to referenced systems

Method	Rock	Classical	Rabi	Pop	Ghazal
Monkey"s Audio [27]	54.15	54.28	54.06	53.67	54.21
WavPack [37]	52.09	51.51	52.19	52.25	52.46
FLAC [5]	53.39	53.26	53.19	52.09	53.78
Proposed method	56.04	56.04	56.01	56.47	56.13

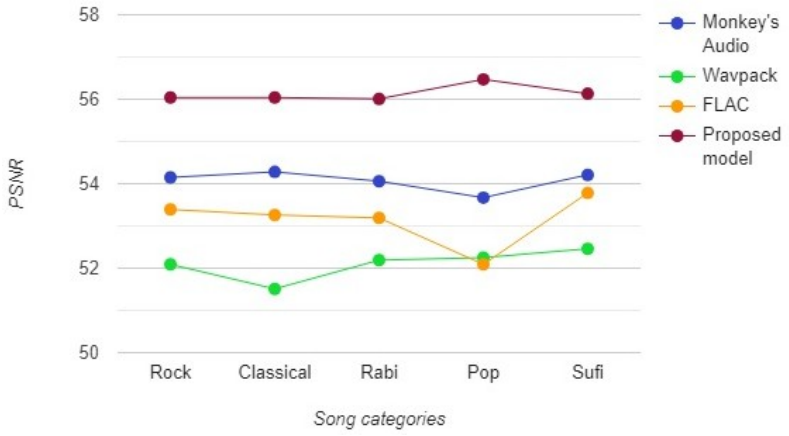

Figure 5. PSNR value comparison in graphic form

The suggested framework has an average entropy score of 13.87, which is higher than the average values of Wavpack Lossless, FLAC, and Monkey"s Audio, which are

13.42, 13.56 and 13.45, respectively. Table 9 shows the evaluated entropy values for the current and previous lossless audio encoding techniques.

Table 9
Aeverage entropy comparison with respect to referenced systems

Metrics	Monkey's Audio [27]	WavPack [37]	FLAC [5]	Proposed method
Entropy (average)	13.45	13.42	13.56	13.87

Table 10 displays the category-wise average entropy calculated for the suggested framework using three cited systems. It is evident from the experiment result set displayed in Table 10 that the current model yields higher entropy in every category. The proposed system's categorywise entropy comparison is displayed in Figure 6.

Table 10
Groupwise entropy comparison of the proposed model with respect to referenced systems

Method	Rock	Classical	Rabi	Pop	Ghazal
Monkey's Audio [27]	13.85	13.54	13.09	13.47	13.34
WavPack [37]	13.56	13.62	13.52	13.28	13.12
FLAC [5]	13.21	13.39	13.49	13.91	13.84
Proposed model	13.89	13.69	13.98	14.24	13.86

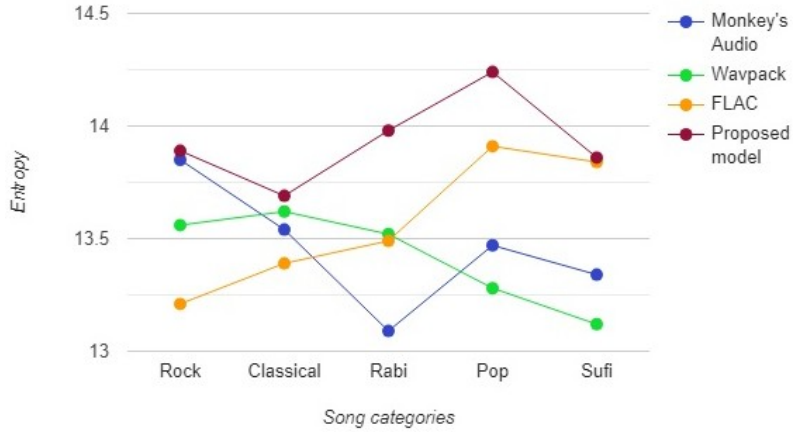

Figure 6. Graphical comparison of the categorywise entropy

The proposed technique's assessed subjective metric MOS of 5 is adequate to demonstrate the excellent caliber of the model's audio reconstruction.

Figure 7 represents the original signal and regenerated audio signal by the model. Therefore, it is visible that regenerated signal is like original signal with negligible deviations.

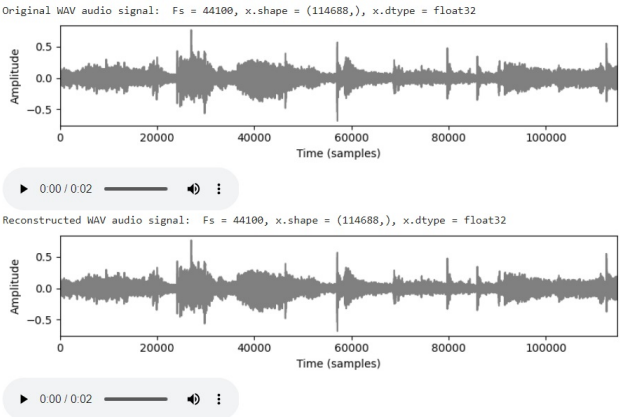

Figure 7. The original and regenerated signals are compared

We tested the suggested model's resilience and performance with three additional deep learning-based lossless audio codecs that are currently in use: i) Daniela N. Rim et al. [33] ii) DLLAE [25] iii) LINNE [22]. The suggested model can reproduce the original audio signal with incredibly little variations. The estimated mean square error value for the suggested model is 0.001713. Furthermore, the suggested model's RMSE is 0.032031. The estimated MAE value for the suggested model is 0.031803. The system's resilience and the degree of similarity between the original and anticipated signals are shown by MAE and RMSE values that are near to 0. Additionally, 0.980152 is another metric known as NCC that is used to gauge the quality of the regenerated signal. Good regeneration is indicated by an NCC value closer to 1. To illustrate the accuracy and resilience of the model, Table presents the evaluated values of the parameters, such as MSE [20] RMSE [38], MAE [38], and NCC, of the current model in comparison to the other mentioned DNN system. The compression ratio (%) generated by the new method is compared with the other DNN model in Table 11. Table 12 indicates that compared to the current models, the suggested lossless audio codec produces higher compression. A visual comparison of category-wise compression for the current method and the models in use is shown in Figure. The comparison of the proposed model's compression ratio (%) with other neural network-based models already in use is displayed in Figure 8.

Table 11
Robustness performance comparison with exiting DNN model

Method	MSE	RMSE	MAE	NCC
Proposed technique	0.001713	0.032031	0.031803	0.985152
DLLAE [25]	0.017691	0.131575	1.153562	0.981212
Daniela N. Rim et al. [33]	0.123475	2.014721	2.526123	0.981042
LINNE [22]	0.134251	0.157514	2.016123	0.980467

Table 12
Categorywise compression comparison

Method	Pop	Sufi	Ghazal	Rabi	Classical
DLLAE [25]	86.18	85.47	86.01	86.01	85.12
Daniela N. Rim et al. [33]	84.21	83.34	83.25	85.81	84.17
LINNE [22]	74.56	74.12	73.38	72.33	72.38
Proposed method	89.04	88.41	90.01	89.67	89.13

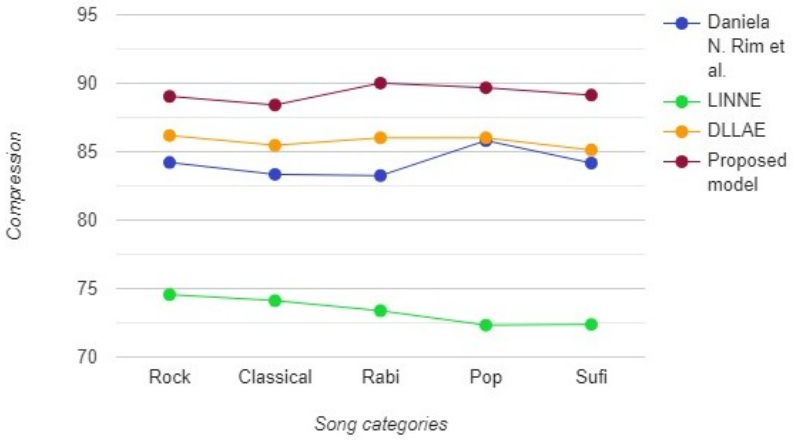

Figure 8. Graphical comparison of the categorywise compression

The experiment measures the audio quality of the reconstructed signal using SDR [33]. Decibel (dB) is used for SDR. SDR is a measure of the reconstructed signal's similarity to the original signal $S_{original}$. It is computed as follows:

$$SDR = 10log_{10}\frac{|S_{original\ signal}|^2}{|S_{reconstructed\ signal} - S_{original\ signal}|^2} \tag{8}$$

The proposed method's signal to distortion ratio (SDR)(dB) is computed and contrasted with the DNN-based audio compression model that is currently in use [6]. The SDR attained by the model is displayed in Table 13 together with the current audio compression model. Table 13 shows that the existing deep neural network model achieves perfect reconstruction, whereas the proposed audio codec produces higher SDR in each of the five song categories. On the other hand, a lower SDR for the current model indicates a considerable loss of audio quality and data. Figure 9 displays a graphical comparison of the categorywise SDR(dB) for the current with the present DNN model.

Table 13
Groupwise SDR comparison

Method	Rock	Classical	Rabi	Pop	Ghazal
Daniela N. Rim et al [33]	24.56	29.01	30.89	24.67	26.12
LINNE [22]	26.46	23.11	25.29	26.17	25.12
DLLAE [25]	24.53	25.07	26.79	24.66	27.22
Proposed model	57.59	58.6	58.69	61.78	60.1

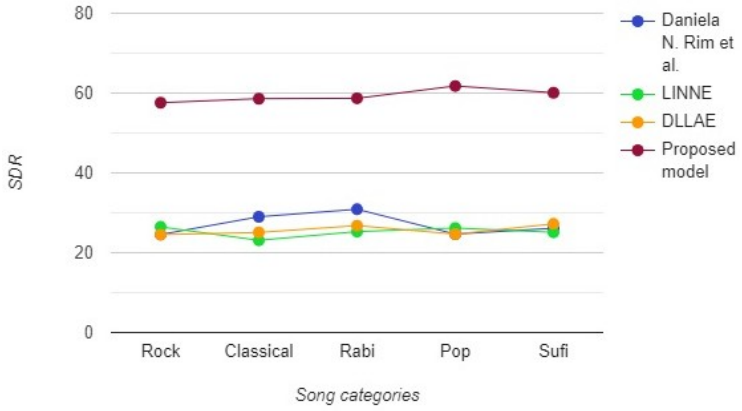

Figure 9. Graphical comparison of SDR

The quality of the regenerated audio measured with MOS which dispalyed in Table 14. Table 14 states that an audio quality grade of "5" denotes "Excellent" sound, while a grade of "1" denotes "Bad" quality. For the current quality measurement work, the ITUR Rec. 500 quality rating is suitable. since it provides a quality rating between 1 and 5 [2]. Since it is over the threshold level of human perception, the MOS (Mean opinion score) value for the current technique, which is 5, suggests that the reconstructed audio quality stays unaffected by this slight data change during the transformation.

Table 14
A scale for assessing the deterioration of audio quality

Rating	Impairment	Quality
5	Imperceptible	Excellent
4	Perceptible, not annoying	Good
3	Slightly annoying	Fair
2	Annoying	Poor
1	Very annoying	Bad

4.2. Comparative study and discussion

The proposed CNN model is based on the encoder and decoder network. Using pooling layers, the decoder upsamples each input that the encoder network receives. Each decoder executes a non-linear up-sampling in order to produce full feature maps from the sparse max-pooling indices produced at each pooling layer in the encoder network. The encoder takes pixel value inputs from the training dataset, and the decoder uses sparse max-pooling indices from encoder pooling layers. In order to facilitate end-to-end training, the encoder decoder design offers a notable improvement in terms of reducing the number of trainable parameters. The primary feature of the suggested architecture is its simplicity in training and easy customization. The encoder generates a latent space representation with low resolution, while the decoder uses the trainable filters to convolve them and provide dense feature maps.

The proposed model's primary distinguishing characteristics are:

- Enables the encoder and decoder networks to be trained using a simple training technique.
- Utilizing the latest activation function, it maximizes potential performance.
- This offers an adaptable framework to modify inputs of any magnitude.

For the proposed CNN model, the input and output shapes of each layer together with the associated parameters are displayed in Figures 10 and 11.

Model: "CNN_encoder_model"

Layer (type)	Output Shape	Param #
input_3 (InputLayer)	[(None, 20, 20, 1)]	0
CNN_Encoder_conv2d_layer1 (Conv2D)	(None, 10, 10, 32)	320
CNN_Encoder_conv2d_layer2 (Conv2D)	(None, 5, 5, 16)	4624
CNN_Encoder_conv2d_layer3 (Conv2D)	(None, 3, 3, 8)	1160
CNN_Encoder_conv2d_layer4 (Conv2D)	(None, 2, 2, 4)	292
CNN_encoded_Latent_space (Conv2D)	(None, 2, 2, 4)	148

Total params: 6544 (25.56 KB)
Trainable params: 6544 (25.56 KB)
Non-trainable params: 0 (0.00 Byte)

Figure 10. CNN encoder model

Model: "CNN_decoder_model"

Layer (type)	Output Shape	Param #
input_4 (InputLayer)	[(None, 4, 4, 4)]	0
CNN_ecoder_con2d_layer2 (Conv2D)	(None, 4, 4, 8)	296
up_sampling2d_5 (UpSampling2D)	(None, 8, 8, 8)	0
CNN_Decoder_con2d_layer3 (Conv2D)	(None, 6, 6, 16)	1168
up_sampling2d_6 (UpSampling2D)	(None, 12, 12, 16)	0
CNN_Decoder_con2d_layer4 (Conv2D)	(None, 10, 10, 32)	4640
up_sampling2d_7 (UpSampling2D)	(None, 20, 20, 32)	0
CNN_Decoder_con2d_Recons_layer (Conv2D)	(None, 20, 20, 1)	289

Total params: 6393 (24.97 KB)
Trainable params: 6393 (24.97 KB)
Non-trainable params: 0 (0.00 Byte)

Figure 11. CNN decoder model

The experiment involves testing various learning rate, batch size, and combinations of layers computing the associated MSE. The optimal parameters selected for the CNN model where the MSE is least are displayed in Table 15. This proposed CNN model has a mean square error loss of 0.001604, which is quite low.

Table 15
CNN Model configuration parameters

Parameter name	Value
Learning rate	0.002
Optimizer	Adam
Inner layer activation function	ReLU
Final layer activation function	Sigmoid
Batch Size	128
Epoch	200
MSE loss	0.001604
No of hidden layer	4

In Figure 12 and 13, for the CNN model, the training progress and corresponding loss function are shown. Figure 13 shows the epoch-wise loss of the model.

Two of the more prevalent standard prediction benchmarks, the Lasso regression and the Ridge regression [31], were used to evaluate the efficacy of the proposed CNN model. With a much smaller MSE loss, the convolutional encoder decoder model faithfully recreates the original audio data. A mean square error value of 0.0.001604 has been assessed for the proposed model. On the other hand, an MAE value of 0.042809 has been established for the proposed framework. The system is robust and the expected and original signals are fairly equivalent, as indicated by the MAE and MSE values that are close to zero. The NCC parameter, with a value of 0.9877651, is another way to measure the quality of the regenerated signal. The NCC value being near to 1 indicates that the CNN model can regenerate competently. Table 16 displays the three models' evaluated data.

```
Epoch 95/100
85/85 [==============================] - 5s 61ms/step - loss: 0.0017 - val_loss: 0.0018
Epoch 96/100
85/85 [==============================] - 5s 55ms/step - loss: 0.0017 - val_loss: 0.0018
Epoch 97/100
85/85 [==============================] - 5s 54ms/step - loss: 0.0017 - val_loss: 0.0018
Epoch 98/100
85/85 [==============================] - 4s 44ms/step - loss: 0.0017 - val_loss: 0.0018
Epoch 99/100
85/85 [==============================] - 5s 59ms/step - loss: 0.0017 - val_loss: 0.0019
Epoch 100/100
85/85 [==============================] - 6s 76ms/step - loss: 0.0017 - val_loss: 0.0018
Running prediction..
21/21 [==============================] - 2s 21ms/step
Plotting results..
metrics name is ['loss']
```

Figure 12. Epochwise training progress

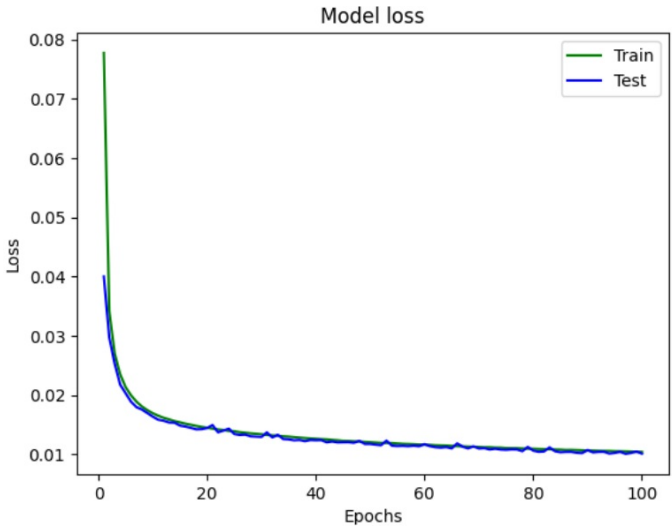

Figure 13. Model loss

Table 16
Performance comparison of the proposed model with other deep learning models

Model	MSE	MAE	NCC
Ridge [31]	0.137621	2.051241	0.964532
Lasso [31]	0.110457	1.112724	0.972153
Proposed technique	0.001604	0.042809	0.9877651

The input audio signal in this proposed audio compression technique undergoes a sequential application of Discrete Cosine Transform (DCT) and weighted tree binary encoding, followed by transformation into non-overlapping images sized at 20×20, which facilitates Convolutional Neural Network (CNN) model training. Utilizing this CNN model further compresses images of size 2×2 by employing dimensionality reduction techniques, seamlessly integrating image compression with audio compression. The experiment compares the suggested framework against state-of-the-art lossless audio compression techniques such as FLAC, Wavpack, and Monkey's audio. Furthermore, three conventional DNN-based audio compression models have been compared with the suggested model [22, 25, 33]. Consequently, it is evident from the experimental data set that the new technique achieved a higher compression ratio with good audio regeneration quality when compared to the current standard audio compression techniques.

5. Conclusion and future scope

The proposed lossless audio codec combines traditional techniques with machine learning-based convolutional neural network (CNN) technology. By sequentially uti-

lizing CNN architecture, binary transformation, and adaptive discrete cosine transform (DCT) encoding, this codec achieves a compression ratio of over 80%. Various metrics are used to evaluate its efficiency, showing that it outperforms other methods. Improvements in PSNR, MOS, SDR and NCC ensure that audio quality remains consistent. Unlike previous models, the current CNN model is easier to comprehend and train. Consequently, this approach provides greater compression than existing lossless audio codecs, maintaining audio quality. The compression that the proposed method achieves is not yet as high as what MP3 achieves, which is above 90%. However, the suggested technique's achieved 89.25% compression is not too far from that of a lossy codec like as MP3. In the future, the CNN and XOR models can have more layers added to them, and the hyperparameters can be trained using more processing power to achieve higher compression performance.

Statements and declarations

Conflicting interests

All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript.

Authorship contributions

The both authors confirm the responsibility for the following: study conception and design, data collection, analysis and interpretation of results, and manuscript preparation.

Data availability statements

Data sharing not applicable to this article as no datasets were generated or analysed during the current study.

Funding

No funds, grants, or other support was received.

References

[1] Apple Lossless Audio Codec, https://macosforge.github.io/alac/. Accessed: 29.06.2024.

[2] Arnold M.: Audio watermarking: Features, applications and algorithms. In: *2000 IEEE International Conference on Multimedia and Expo. ICME2000. Proceedings. Latest Advances in the Fast Changing World of Multimedia (Cat. No.00TH8532)*, vol. 2, pp. 1013–1016, IEEE, 2000. doi: 10.1109/icme.2000.871531.

[3] Audacity, https://www.audacityteam.org/. Accessed: 29.06.2024.

[4] Chęiński K., Wawrzyński P.: DCT-Conv: Coding filters in convolutional networks with Discrete Cosine Transform. In: *2020 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2020. doi: 10.1109/ijcnn48605.2020.9207103.

[5] Coalson J.: "FLAC: Free lossless audio codec", Xiph. Org Foundation, https://xiph.org/flac/index.html. Accessed: 30.01.2024.

[6] Debnath A., Mondal U.K.: Lossless audio codec based on CNN, weighted tree and arithmetic encoding (LACCWA), *Multimedia Tools and Applications*, vol. 83, pp. 48737–48759, 2024. doi: 10.1007/s11042-023-17393-4.

[7] Debnath A., Mondal U.K., Roy B.B., Panja N.: Achieving lossless audio encoder through integrated approaches of wavelet transform, quantization and Huffman encoding (LAEIWQH). In: *2020 International Conference on Computer Science, Engineering and Applications (ICCSEA)*, IEEE, 2020. doi: 10.1109/iccsea49143.2020.9132865.

[8] Diniz P.S.R.: *Adaptive filtering. Algorithms and Practical Implementation*, Springer New York, NY, 1997. doi: 10.1007/978-1-4419-8660-3.

[9] Freitag M., Amiriparian S., Pugachevskiy S., Cummins N., Schuller B.: auDeep: Unsupervised learning of representations from audio with deep recurrent neural networks, *The Journal of Machine Learning Research*, vol. 18(1), pp. 6340–6344, 2017.

[10] Gao W., Huang T., Reader C., Dou W., Chen X.: IEEE standards for advanced audio and video coding in emerging applications, *Computer*, vol. 47(5), pp. 81–83, 2014. doi: 10.1109/mc.2014.122.

[11] Gersho A.: Adaptive filtering with binary reinforcement, *IEEE Transactions on Information Theory*, vol. 30(2), pp. 191–199, 1984. doi: 10.1109/tit.1984.1056890.

[12] Ghido F., Tabus I.: Sparse modeling for lossless audio compression, *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21(1), pp. 14–28, 2012. doi: 10.1109/tasl.2012.2211014.

[13] Gunawan T.S., Zain M.K.M., Muin F.A., Kartiwi M.: Investigation of lossless audio compression using ieee 1857.2 advanced audio coding, *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 6(2), pp. 422–430, 2017. doi: 10.11591/ijeecs.v6.i2.pp422-430.

[14] Gupta M., Garg A.K.: Analysis of image compression algorithm using DCT, *International Journal of Engineering Research and Applications (IJERA)*, vol. 2(1), pp. 515–521, 2012.

[15] Huang H., Shu H., Yu R.: Lossless audio compression in the new IEEE standard for advanced audio coding. In: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6934–6938, IEEE, 2014. doi: 10.1109/icassp.2014.6854944.

[16] Jadhav S., Patole R., Rege P.: Audio splicing detection using convolutional neural network. In: *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, IEEE, 2019. doi: 10.1109/icccnt45670.2019.8944345.

[17] Kameoka H., Kamamoto Y., Harada N., Moriya T.: A linear predictive coding algorithm minimizing the Golomb-Rice code length of the residual signal, *IEICE Transactions on Fundamentals of Electronics*, vol. 91(11), pp. 1017–1025, 2008.

[18] Kankanahalli S.: End-to-end optimized speech coding with deep neural networks. In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2521–2525, IEEE, 2018. doi: 10.1109/icassp.2018.8461487.

[19] Lin Y.C., Hsu Y.T., Fu S.W., Tsao Y., Kuo T.W.: IA-NET: Acceleration and Compression of Speech Enhancement Using Integer-Adder Deep Neural Network. In: *Interspeech*, pp. 1801–1805, 2019. doi: 10.21437/interspeech.2019-1207.

[20] Manju M., Abarna P., Akila U., Yamini S.: Peak signal to noise ratio & mean square error calculation for various images using the lossless image compression in CCSDS algorithm, *International Journal of Pure and Applied Mathematics*, vol. 119(12), pp. 14471–14477, 2018.

[21] Mineo T., Shouno H.: Improving sign-algorithm convergence rate using natural gradient for lossless audio compression, *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2022, 12, 2022. doi: 10.1186/s13636-022-00243-w.

[22] Mineo T., Shouno H.: A lossless audio codec based on hierarchical residual prediction. In: *2022 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pp. 123–130, IEEE, 2022. doi: 10.23919/apsipaasc55919.2022.9980327.

[23] Mondal U.K., Debnath A.: Developing a dynamic cluster quantization based lossless audio compression (DCQLAC), *Multimedia Tools and Applications*, vol. 80(6), pp. 8257–8280, 2021. doi: 10.1007/s11042-020-09886-3.

[24] Mondal U.K., Debnath A.: Designing a novel lossless audio compression technique with the help of optimized graph traversal (LACOGT), *Multimedia Tools and Applications*, vol. 81(28), pp. 40385–40411, 2022. doi: 10.1007/s11042-022-12556-1.

[25] Mondal U.K., Debnath A., Mandal J.K.: Deep learning-based lossless audio encoder (DLLAE), *Intelligent Computing: Image Processing Based Applications*, pp. 91–101, 2020. doi: 10.1007/978-981-15-4288-6_6.

[26] Mondal U.K., Debnath A., Tabassum N., Mandal J.K.: Designing an Iterative Adaptive Arithmetic Coding-Based Lossless Bio-signal Compression for Online Patient Monitoring System (IAALBC). In: J.K. Mandal, D. De (eds.), *Frontiers of ICT in Healthcare, Proceedings of EAIT 2022,* Lecture Notes in Networks and Systems, vol. 519, pp. 655–664, Springer, 2023. doi: 10.1007/978-981-19-5191-6_53.

[27] Monkey's Audio. A fast powerful lossless audio compressor, https://monkeysaudio.com/index.html. Accessed: 30.01.2024.

[28] Moriya T., Harada N., Kamamoto Y., Sekigawa H.: MPEG-4 ALS international standard for lossless audio coding, *NTT Technical Review*, vol. 4(8), pp. 40–45, 2006.

[29] Nowak N., Zabierowski W.: Methods of sound data compression–comparison of different standards, *Radio Electronics and Informatics*, No. 4, 2011.

[30] Patil M., Gupta A., Varma A., Salil S.: Audio and speech compression using DCT and DWT techniques, *International Journal of Innovative Research in Science, Engineering and Technology*, vol. 2(5), pp. 1712–1719, 2013.

[31] Pedro H.T., Larson D.P., Coimbra C.F.M.: A comprehensive dataset for the accelerated development and benchmarking of solar forecasting methods, *Journal of Renewable and Sustainable Energy*, vol. 11(3), 036102, 2019. doi: 10.1063/1.5094494.

[32] Reznik Y.A.: Coding of prediction residual in MPEG-4 standard for lossless audio coding (MPEG-4 ALS). In: *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 3, pp. 1024–1027, 2004. doi: 10.1109/ICASSP.2004.1326722.

[33] Rim D.N., Jang I., Choi H.: Deep neural networks and end-to-end learning for audio compression, *Journal of KIISE*, vol. 48(8), pp. 940–947, 2021. doi: 10.5626/jok.2021.48.8.940.

[34] Shukla S., Ahirwar M., Gupta R., Jain S., Rajput D.S.: Audio compression algorithm using discrete cosine transform (DCT) and Lempel-Ziv-Welch (LZW) encoding method. In: *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, pp. 476–480, IEEE, 2019. doi: 10.1109/comitcon.2019.8862228.

[35] Tau Software Projects, http://tausoft.org/wiki/True_Audio_Codec_Overview. Accessed: 29.06.2024.

[36] Tensorflow – Keras, https://www.tensorflow.org/api_docs/python/tf/keras. Accessed: 29.06.2024.

[37] WavPack. Hybrid Lossless Audio Compression, http://www.wavpack.com. Accessed: 29.06.2024.

[38] Willmott C.J., Matsuura K.: Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance, *Climate Research*, vol. 30(1), pp. 79–82, 2005. doi: 10.3354/cr030079.

[39] Zhou Y., Wang C., Zhou X.: DCT-Based Color Image Compression Algorithm Using an Efficient Lossless Encoder. In: *2018 14th IEEE International Conference on Signal Processing (ICSP)*, pp. 450–454, 2018. doi: 10.1109/ICSP.2018.8652455.

Affiliations

Asish Debnath
Vidyasagar University Midnapore, Department of Computer Science, 721102, West Bengal, India, debnathasish@gmail.com

Uttam Kr. Mondal
Vidyasagar University Midnapore, Department of Computer Science, 721102, West Bengal, India, uttam_ku_82@yahoo.co.in

Received: 11.07.2024
Revised: 16.09.2024
Accepted: 7.04.2025