Proceedings of Engineering and Technology Innovation, vol. 26, 2024, pp. 45-54

A Codebook Compression Method for Vector Quantization Algorithm

Abul Hasnat[1,*], Dibyendu Barman[1], Md Azizul Hoque[2], Santanu Halder[3], Debotosh Bhattacharjee[4]

[1]Government College of Engineering and Textile Technology, Berhampore, West Bengal, India

[2]Sreegopal Banerjee College, Magra, Hooghly, West Bengal, India

[3]Government College of Engineering and Leather Technology, Kolkata, West Bengal, India

[4]Jadavpur University, Kolkata, West Bengal, India

Received 08 January 2024; received in revised form 21 February 2024; accepted 22 February 2024

DOI: https://doi.org/10.46604/peti.2024.13268

Abstract

This study introduces a novel approach to enhance the compression ratio of the vector quantization (VQ) algorithm by specifically targeting the compression of its codebook. The VQ algorithm typically generates an index matrix and a codebook to represent compressed images. The proposed method focuses on reducing the size of the codebook, which comprises N codewords, each with elements quantized into four levels. Each 8-bit element in a codeword is compressed to 2-bits, and the encoded codeword is accompanied by the minimum value and a threshold value in the codebook. Experimental results on benchmark color images, such as baboon, airplane, Lena, and others, demonstrate a significant reduction of 62.50% in the size of the VQ codebook.

Keywords: codebook, peak signal-to-noise ratio, structural similarity index, quantization, VQ

1. Introduction

The vector quantization (VQ) algorithm [1-5] is a lossy image compression technique. It divides an image into image blocks and each such image block forms a training vector. All such training vectors are combined to build the training dataset for the image. Linde-Buzo-Gary (LBG) algorithm is applied to this training dataset to compress the image and as a result, LBG produces one index matrix and one codebook as the compressed form of the original image.

The codebook contains N codewords/ code vectors. The number of code vectors or codewords is 128 or 256 generally. Each training vector is associated with one of the code vectors. The association between the training vectors and codewords is specified by an index matrix indicating which training vector maps to which code vector. The visual quality of the compressed image using the VQ depends on the selection of the right set of codewords (designing the optimal codebook).

The size of the compressed image using VQ is the total size [1-2] of the codebook and the size of the index matrix. The compression ratio of the algorithm is in the approximate range of 88% to 95% [1] for a standard color image (three-channel) of size 256 × 256 to 512 × 512 with a codebook size of 256 × 16. The compression ratio of the VQ (for a predefined size of the codebook) is not scalable [1] for an image.

This paper is divided into five sections. Section 2 gives a brief discussion of the literature survey. Section 3 discusses the codebook compression method. Section 4 presents the experimental findings and a discussion of them. The paper is concluded in Section 5.

* Corresponding author. E-mail address: email.abulhasnat@gmail.com

English language proofreader: Chih-Wen Teng

2. Literature Review

In the literature, a few studies are reported on the modification of the codebook of VQ. These methods are covered briefly below. Some studies focused on the learning of the codebook of VQ using machine learning [4] and convolutional neural networks (CNN) [5]. Another approach reported the usage of evolutionary algorithms (EA) [6-8] for better codebook design. As input, they have taken the LBG-generated codebooks (of multiple runs) as initial search points and then applied genetic algorithm (GA), particle swarm optimization (PSO), whale optimization, etc., algorithms to optimize the codebook further.

In 2016, Shah et al. [9] reported a hybrid technique for codebook and index matrix compression of the VQ algorithm. They dropped elements of odd positions of a code vector (of size 16) to get a new code vector of size 8 for codebook compression. Also, they used the search order coding (SOC) concept, for index matrix compression. They further reduced the size of the VQ compressed image. However, the visual quality of the decompressed image is degraded significantly.

Rahebi [10] 2022 reported an efficient codebook generation using EA for image compression. The codebook is optimized using the whale optimization algorithm. The method focused on producing high-quality reconstructed images using the optimized codebook. The method [10] reported more efficient compression in terms of image quality than the evolutionary algorithms such as particle swarm optimization, bat, and firefly algorithms. The study is not about improving the compression ratio of the VQ.

In 2022, Barman et al. [11] reported a codebook updating procedure to enhance the VQ method's performance. This method reduces the size of the codebook by 37.50% and improves the compression ratio of the algorithm by representing each element in the codebook with 5-bits rather than 8-bits. But using this method, the size of the codebook is reduced by 37.50% only.

Again, in 2022, Barman et al. [12] developed another codebook update method that focused on the improvement of the visual quality of the decompressed image. It is applied to multiple images together. During the compression phase, the method is applied to multiple codebooks of more than one image at the same time. It divides the codebook into two parts. The first part of the codebook contains the code vectors with high frequencies and it is kept unchanged. But the second part of the codebook containing code vectors with low frequencies is further compressed. Here code vectors are converted into code vectors of four levels using quantization for better compression ratio. Also in a few cases, the visual quality of the reconstructed image is affected to some extent.

In 2023, Chavan et al. [13] reported an analysis of codebook optimization for image compression using EA. This study examines the codebook optimization of VQ using the modified GA, and PSO. They concluded that the PSO performs better in optimizing the codewords for the training samples for image compression. However, the initial selection approach is important in the PSO algorithm.

Most of the studies focused on better codebook design. Shah et al. [9] compressed the VQ-generated codebook and index matrix further but the visual quality of the compressed image is affected reasonably. Also, Barman et al. [12] reported the method where they developed a common codebook for multiple images. They further compressed the second part of the codebook containing the codewords with low frequencies.

The objective of this study is to improve the compression ratio of the VQ algorithm. VQ produces an index matrix and a codebook as the compressed form of an image. The size of the compressed image using VQ is the sum of the size of the index matrix and the size of the codebook. The contribution of this study is that the VQ-generated codebook is compressed further. As a result, the size of the codebook is reduced by 62.50%. Overall compression performance of the VQ method is further improved. At the same time, the visual quality of the reconstructed image using this codebook compression method is almost similar to the visual quality of the reconstructed image using the VQ. The method is tested extensively on benchmark color images and color images of UCID.v2 database [14].

3. Method

Image compression using VQ takes an image as input, forms the training vector, and then applies the LBG algorithm to the training dataset. The steps of the VQ [8] algorithm are explained below.

Step 1: VQ divides an image into $p \times p$ blocks (sub-images). Each block forms one training vector of size k where $k = p \times p$. All these training vectors of the entire image form the training vector set. A training vector, $X = (x_1, x_2, x_3, \cdots, x_k)$ is a k-dimensional array. Let there be an M number of such training vectors in the training dataset.

Step 2: Generate a codebook containing N ($< M$) number of code words. Each code word/vector is of length k. Generally, the codebook is generated using the LBG algorithm. Let the codebook be a set of N vectors, $CB = \{C_1, C_2, C_3, \cdots, C_N\}$. Where C_i contains k number of elements. The codebook contains the representatives of all the training vectors. Every training vector must be mapped into one of the code vectors. This association is decided based on the minimum distance between the code vector and the training vector. Thus, there is an index matrix for indicating specific training vector mapping to a specific code vector. It contains one index value for each $p \times p$ sub-image/block. The objective of the LBG algorithm is to find the codebook for which the total distortion during the quantization of training vectors is minimized.

The LBG algorithm produces one codebook and one index matrix. The generated codebook along with the index matrix is stored as a compressed form of the original image. Application of a lossy compression on the index matrix is not desirable because if the index value is altered, then it may point to a wrong code vector resulting distorted reconstructed image. So, the study aims to compress the codebook for better a compression ratio. The goal of the study is to compress the VQ-generated codebook further. Let the size of the VQ-generated a codebook $N \times 16$. The codebook consists of N number of codewords and each codeword consists of sixteen elements. Fig. 1 shows one sample codework containing sixteen elements. Each element has a depth of 8-bits. Generally, N is taken as 256.

23	36	52	48	36	58	46	62	16	58	49	62	66	37	43	48

Fig. 1 A sample codeword of size 16

This section explains the method to compress the codebook further for a better compression ratio. Thus, if the method is integrated along with VQ, then the VQ becomes a two-step compression method- (1) VQ compression and (2) codebook compression. Here, the second step is explained. It is codebook compression and codebook decompression methods. The steps of the encoding and decoding process are explained below.

3.1. Encoding process

The steps of encoding the VQ-generated codebook are given in Algorithm 1.

Algorithm 1: Codebook compression

Input: VQ-generated codebook, **Output:** Compressed codebook

Step 1: Let the *i*th codeword is cw_i. Here minimum and maximum values of i^{th} codeword, cw_i are $MIN_i = MIN\{cw_i\}$ and $MAX_i = MAX\{cw_i\}$. The threshold value, T_i for i^{th} codeword is also calculated using:

$$T_i = \frac{MAX_i - MIN_i}{3} \tag{1}$$

For the codeword shown in Fig.1, the minimum value, $MIN_i = 16$, maximum value, $MAX_i = 66$, and threshold value, $T_i = \frac{66-16}{3} = (66-16)/3 = 17$. Fig. 2 shows the MIN_i and T_i values of the codeword shown in Fig. 1. For each codeword, the minimum MIN_i and threshold T_i values are stored along with the encoded codeword.

16	17

Fig. 2 The minimum and threshold values of the codeword shown in Fig. 1

Step 2: The range, $MAX_i - MIN_i$ is quantized into four levels as follows. There is a tradeoff between the number of levels considered and the information loss. Increasing the number of levels reduces information loss but demands more storage space for each element. Conversely, decreasing the number of levels reduces storage space but may result in higher information loss. The goal is to identify the optimized number of levels for a code vector, balancing minimal information loss with high compression efficiency. Selecting two levels (bit depth 1) incurs significant information loss while opting for eight levels (bit depth 3) increases storage requirements. So, four levels (bit depth 2) are considered in this study for optimized results.

$$L_0 = MIN_i \tag{2}$$

$$L_1 = MIN_i + T_i \tag{3}$$

$$L_2 = MIN_i + 2 \times T_i \tag{4}$$

$$L_3 = MIN_i + 3 \times T_i \tag{5}$$

So, for the codeword shown in Fig. 1, the quantized four levels are $L_0 = 16$, $L_1 = 33$, $L_2 = 50$, and $L_3 = 67$.

Step 3: For each element in the codeword, the Euclidean distance between the element and all four levels L_0, L_1, L_2, and L_3 are calculated. Each element is replaced with that level index which has the minimum distance.

0	1	2	2	1	2	2	3	0	2	2	3	3	1	2	2

Fig. 3 The quantized levels of the codewords (shown in Fig. 1)

Step 4: Every value of the newly created codeword is converted into a 2-bit binary value. Each element shown in Fig. 3 is converted into a 2-bit binary pattern and it is shown in Fig. 4.

00	01	10	10	01	10	10	11	00	10	10	11	11	01	10	10

Fig. 4 The generated codeword once each value is converted to a 2-bit binary value

Step 5: Four consecutive binary values (as shown in Fig. 4) are concatenated together. Fig. 5 shows the concatenated binary string of 8-bits for the four consecutive 2-bit binary strings shown in Fig. 4.

00 01 10 10	01 10 10 11	00 10 10 11	11 01 10 10

Fig. 5 Merged binary codeword after taking each set of four consecutive values together (from Fig. 4)

Step 6: Convert each eight consecutive binary bits into decimal. Fig. 6 shows the converted decimal values of the binary strings in Fig. 5.

26	107	43	218

Fig. 6 Decimal codeword

This process is repeated for every code vector of the VQ-generated codebook to get the compressed codebook. A portion of the codebook generated by the VQ algorithm for the R channel of the Pepper image is shown in Table 1. Table 2 shows the encoded codebook of the VQ-generated codebook shown in Table 1. The second last column and the last column of Table 2 show the minimum and threshold values (of the respective codebook in Table 1) respectively. This modified codebook as shown in Table 2 is stored as the compressed form of the VQ-generated codebook. The size of the compressed version of the codebook is $N \times (4 + 2) = N \times 6$ instead of the original size $N \times 16$, where each element is of depth 8-bits.

Table 1 A portion of the codebook generated by the VQ algorithm of the R channel of the pepper image

24	22	22	23	21	20	21	24	20	20	19	21	23	21	20	21
85	72	63	61	83	75	73	66	131	128	80	71	133	83	75	73
67	55	62	61	58	54	54	62	55	52	59	71	68	58	54	54
119	119	130	139	117	122	126	138	118	120	129	139	114	117	122	126
60	57	60	82	53	54	54	63	64	53	52	60	72	53	54	54
111	106	121	121	93	95	119	124	88	87	115	132	82	93	95	119

Table 2 Compressed codewords and their respective minimum and threshold values of the codebook in Table 1.

Compressed codewords				Minimum	Threshold
224	87	81	149	19	2
64	84	244	213	61	24
218	66	71	112	52	6
91	27	27	6	114	8
87	2	7	192	52	7
175	95	11	23	87	12

3.2. Decoding of the encoded codebook:

It is the reverse process of the encoding steps. The steps of decoding the encoded codebook are discussed in Algorithm 2.

Algorithm 2: Decoding the encoded codebook

Input: Compressed codebook, **Output:** Decompressed codebook

Step 1: Convert each decimal value of the codeword into an 8-bit binary value. Split the 8-bit binary string into four by taking two consecutive binary bits together. Fig. 7 shows the 8-bit and 2-bit binary strings of the elements shown in Fig. 6.

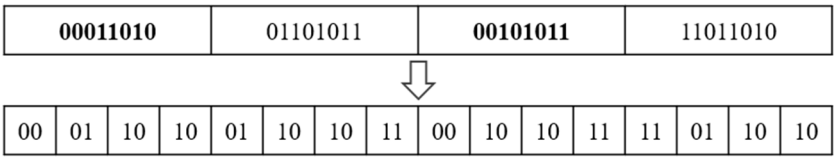


Fig. 7 Codeword after conversion of each decimal value of Fig. 6 into 8-bit binary value then taking 2-bits together

Step 2: Take each 2-bit binary value and convert it into a decimal value. These are the index values of the quantized levels.

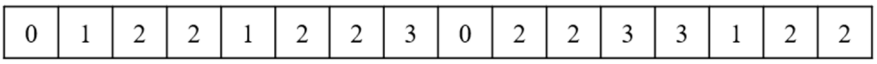


Fig. 8 Codeword after translating each 2-bit binary value of the codeword shown in Fig. 7 to decimal

Step 3: Let the value of a cell be x in Fig. 8. Then the corresponding element or quantized level is calculated as

$$L' = MIN + T \times x \tag{6}$$

where MIN is the minimum value of the codeword and T is the threshold value. MIN and T are stored along with the compressed codeword. Replace elements with the computed level value. Thus, all elements of the codeword are retrieved. The computed elements are shown in Fig. 9 for each element of the Fig. 8.

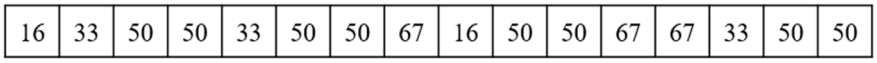


Fig. 9 Retrieved codewords after substituting the corresponding level value for each element in Fig. 8

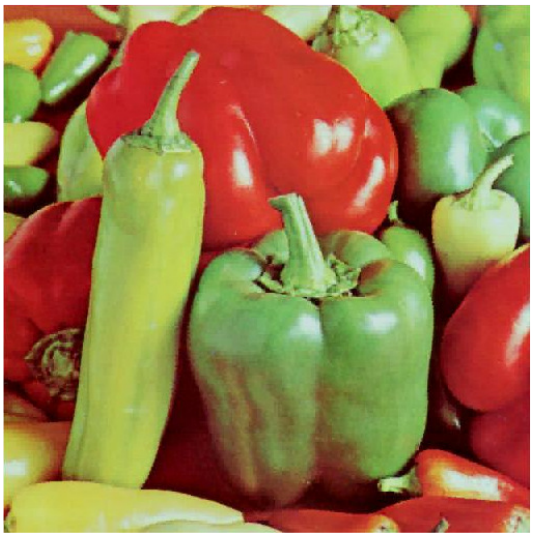
The codeword shown in Fig. 9 is the decoded codeword. To decompress the codebook completely, repeat the decoding process for all N number of codewords.

4. Experimental Result

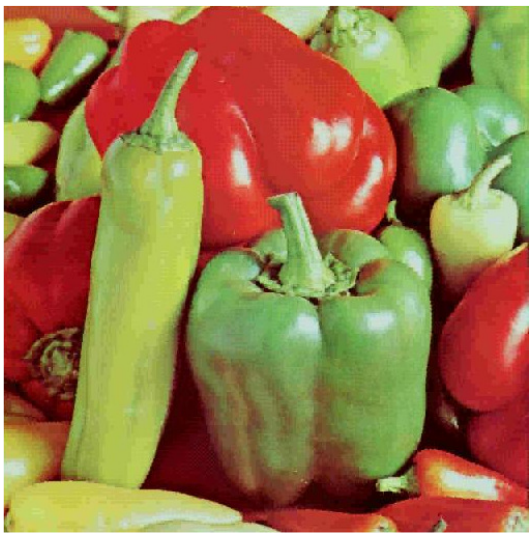
The codebook compression method is tested on benchmark color images i.e.- Lena, baboon, house, etc., and images from the UCID.v2 database. It is implemented using the MATLAB2018. The performance is evaluated using three performance metrics [15-16]- (a) compression ratio, (b) peak signal-to-noise ratio, and (c) structural similarity index

Fig. 10 shows the original pepper image, and the decompressed images, using the conventional VQ, codebook modification method (CBM), and the VQ with codebook compression method respectively. It shows that the visual quality of decompressed images using the VQ and new method (it is a two-step VQ compression method where for the second stage, codebook compression is done by above discussed method) is almost similar. Here, the codebook size is taken as 256 and each codeword contains 16 elements (the block size is 4) for all three channels.

(a) Original pepper image



(b) Compressed image using the VQ



(c) Compressed image using the CBM

(d) Compressed image using VQ with codebook compression

Fig. 10 Original pepper image and its compressed version using different methods

The standard VQ-generated codebook [1-4] is of size $N \times 16$ where $N = 128$ or 256, 16 is the number of elements in each codeword, size of each element is one byte (8-bit depth). Therefore, the required storage space for the codebook is $N \times 16 \times 8 = N \times 128$ bits. Using the new approach, each encoded codeword contains four elements of length 8-bits, also each codeword is required to store the minimum and threshold values. Each of these minimum and threshold values is 8-bits in length. Here, each encoded codeword requires $4 \times 8 + 8 + 8 = 48$ bits. Therefore, to store the compressed codeword requires $N \times 48$ bits. So only for the codebook the required storage space is reduced by $(N \times 128 - N \times 48/N \times 128) \times 100\% =$

62.50%. The method effectively reduces the storage space requirement by up to 3% more than the VQ for an image of size 256×256 to 512×512. Table 3 shows the percentage of storage space reduction achieved using the VQ, CBM, and VQ with codebook compression. Here, the new method performs better than CBM, and also CBM performs better than VQ.

Table 3 Amount of storage space reduced using VQ [1], codebook modification [12], and VQ with codebook compression

Image	Amount of storage space reduced (percentage)		
	VQ [1]	CBM [12]	VQ with codebook compression
4.1.01.tiff	88.48	90.81	91.08
4.1.02.tiff	88.67	91.06	91.28
4.1.03.tiff	90.77	93.22	93.37
4.1.04.tiff	88.39	90.85	90.99
4.1.05.tiff	89.15	91.49	91.75
4.1.06.tiff	89.00	91.46	91.60
4.1.07.tiff	89.43	91.87	92.04
4.1.08.tiff	89.44	91.78	92.04
4.2.07.tiff	93.24	93.81	93.90
mandril.tif	92.72	93.18	93.37
4.2.05.tiff	93.85	94.42	94.50
house.tiff	94.20	94.78	94.85
ucid00006	93.10	93.63	93.96
ucid00007	92.00	92.73	92.86
ucid00008	93.16	93.74	94.03
ucid00028	93.29	94.08	94.15

Fig. 11 shows a bar chart for performance comparison in terms of the percentage of space reduction achieved using VQ, CBM, and VQ with codebook compression. It is shown for sixteen images of varying sizes- benchmark color images of size 256×256, benchmark color images of size 512×512, and four images of UCID.v2.0 of size 384×512. In this respect, it may be observed that the VQ with the codebook compression method performs better than the CBM and VQ. For the index matrix, run-length encoding (RLE) has been applied to all 8×8 blocks in a zigzag manner.

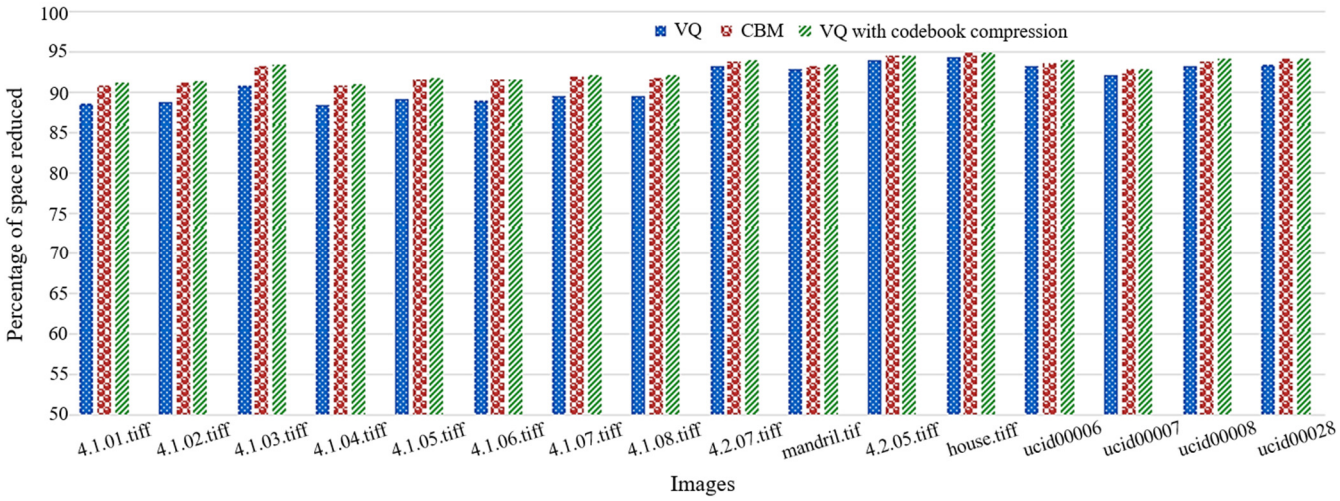


Fig. 11 Percentage of space reduction using VQ, codebook modification, and VQ with codebook compression

The PSNR between the original image and encoded image using VQ, CBM, and the new method is shown in Table 4. It shows that the visual quality of the encoded image using the VQ is slightly better than VQ with codebook compression in terms of PSNR. But it performs better than the CBM. Fig. 12 shows the comparative bar chart of the average PSNR obtained between the original image and encoded image using VQ, CBM, and VQ with codebook compression. It is again shown for the same set of images as shown in Fig. 11. The visual quality of the decompressed images using the VQ is slightly better than the new approach.

Table 4PSNRbetween the original and encoded image using VQ, CBM, and VQ with codebook compression

Image	PSNR								
	VQ			CBM			VQ with codebook compression		
	R	G	B	R	G	B	R	G	B
4.1.01.tiff	34.66	38.39	36.87	32.97	37.45	36.57	34.36	38.41	36.85
4.1.02.tiff	35.31	39.93	38.70	34.88	38.74	37.82	35.10	39.88	38.66
4.1.03.tiff	38.49	46.30	42.80	37.15	45.45	41.89	37.90	46.18	42.68
4.1.04.tiff	36.13	37.62	35.70	35.14	37.12	35.40	35.72	37.54	35.72
4.1.05.tiff	35.52	37.10	35.48	33.86	36.87	34.53	35.06	36.98	35.49
4.1.06.tiff	29.64	36.52	33.42	29.14	36.00	33.12	29.34	36.45	33.40
4.1.07.tiff	39.15	39.42	38.18	37.92	38.98	37.88	38.37	39.52	38.24
4.1.08.tiff	37.06	37.98	37.14	34.49	36.72	36.52	36.32	37.97	37.20
4.2.07.tiff	33.51	35.17	34.73	32.22	34.95	34.45	33.29	35.16	34.74
mandril.tif	25.83	30.25	30.84	25.16	30.79	31.70	25.75	30.22	30.82
4.2.05.tiff	33.35	37.58	37.34	32.78	36.87	36.98	33.06	37.57	37.29
house.tiff	31.21	36.17	33.34	30.78	35.65	33.12	31.04	36.10	33.31
ucid00006	25.82	36.11	35.53	24.95	36.49	35.92	25.74	35.94	35.33
ucid00007	25.06	33.76	34.73	23.91	34.52	35.40	24.87	33.59	34.14
ucid00008	28.34	37.65	37.39	27.29	37.45	37.40	28.17	37.47	37.21
ucid00028	30.37	40.52	39.25	29.58	39.97	38.02	30.04	40.30	39.04

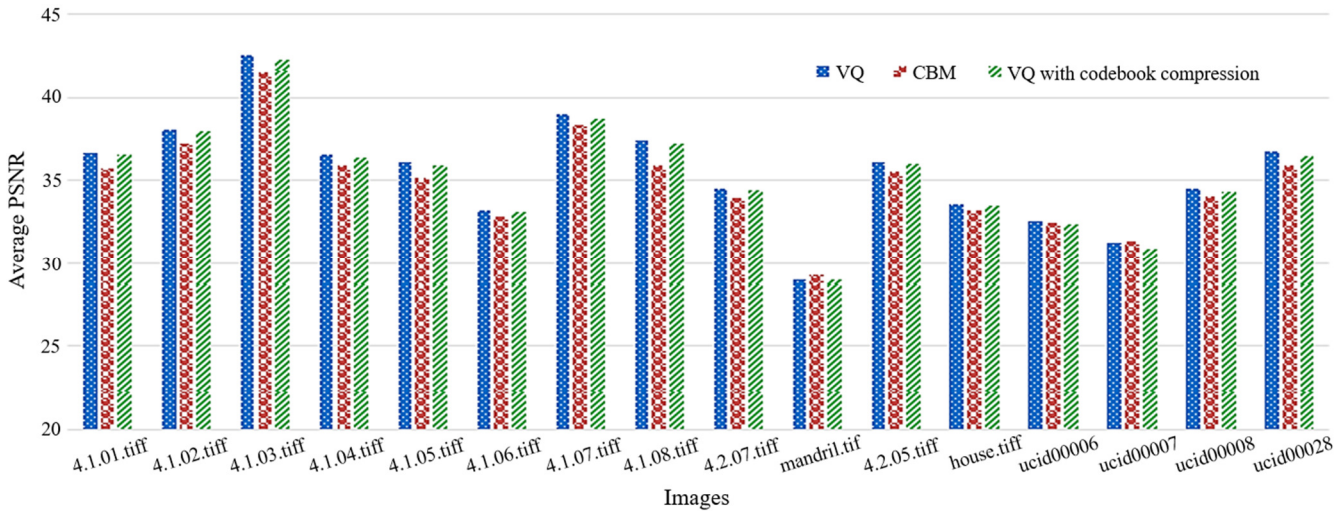


Fig. 12 Average PSNR between the original image and encoded image using VQ, CBM, and VQ with codebook compression

Table 5 shows the SSIM between the original image and the encoded image using VQ, the CBM, and the new method. In terms of SSIM, the visual quality of compressed images using the new method and the VQ is almost similar. Again, it is better than the codebook modification method.

Table 5 SSIM using VQ, Codebook modification, and VQ with codebook compression

Image	SSIM		
	VQ [2]	CBM [13]	VQ with codebook compression
4.1.01.tiff	0.8999	0.8569	0.8966
4.1.02.tiff	0.9080	0.8991	0.9046
4.1.03.tiff	0.9544	0.9487	0.9525
4.1.04.tiff	0.9549	0.9488	0.9532
4.1.05.tiff	0.9595	0.9482	0.9582
4.1.06.tiff	0.9097	0.9042	0.9056
4.1.07.tiff	0.9850	0.9821	0.9844
4.1.08.tiff	0.9824	0.9662	0.9810
4.2.07.tiff	0.9714	0.9657	0.9707
mandril_color.tif	0.8086	0.8020	0.8089

Table 5 SSIM using VQ, Codebook modification, and VQ with codebook compression (continued)

Image	SSIM		
	VQ [2]	CBM [13]	VQ with codebook compression
4.2.05.tiff	0.9140	0.9098	0.9112
house.tiff	0.9246	0.9210	0.9229
ucid00006	0.7934	0.7588	0.7924
ucid00007	0.8585	0.8294	0.8560
ucid00008	0.8758	0.8381	0.8736
ucid00028	0.8912	0.8714	0.8888

Fig. 13 shows the bar chart of SSIM obtained between the original image and the encoded image using VQ, the CBM, and the new method for the same set of images. It is observed that in terms of SSIM, the visual quality for decompressed images using VQ with codebook compression and VQ is almost similar. The new method always performs better than the CBM.

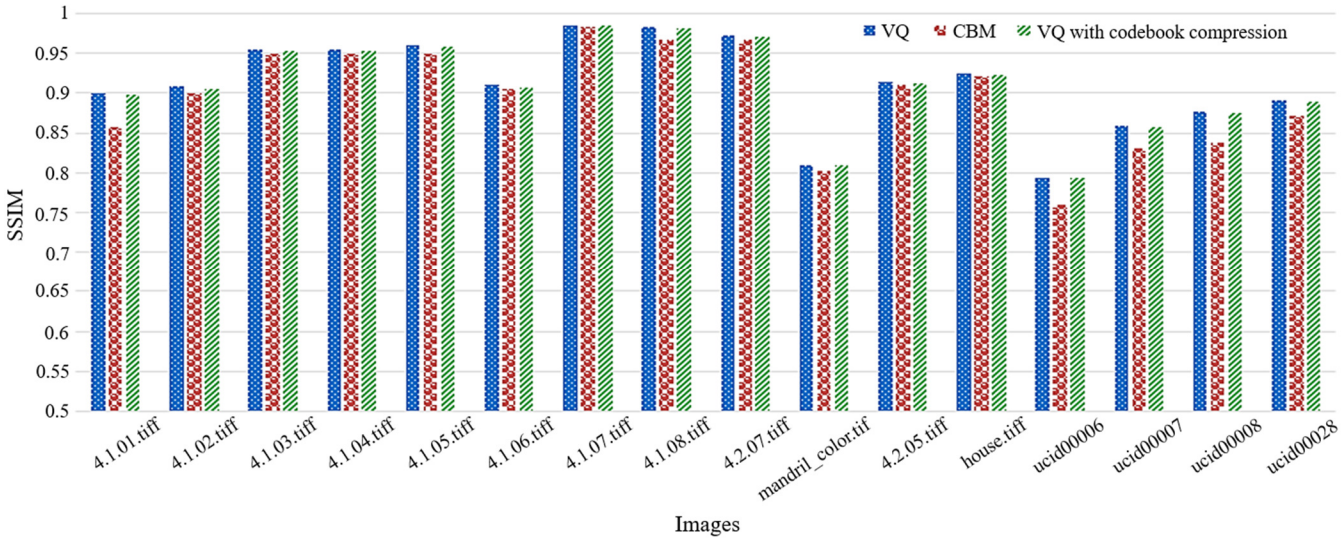


Fig. 13 SSIM between an original image and the encoded image using VQ, Codebook modification, and VQ with codebook compression

Another perspective is the additional time complexity introduced for the VQ due to the compression of the codebook. The time complexity of the codebook compression method $O(N)$, where N is the number of codewords in the codebook. Additional computations for codebook compression have four steps- (a) Searching the minimum and maximum value of the code vector- It requires a fixed number of computations. Because the size of the code vector and the number of code vectors in a codebook are two predetermined fixed values. Generally, the size of the code vector is 16, and the codebook size (number of code vectors) is either 128/256 respectively, (b) Finding the threshold for mapping into four levels- for a given code vector and given maximum and minimum values, a division operation is required to find the threshold value. (c) Quantization step- at most four multiplications and four addition operations are required to quantize each code word into four levels. Thus, the number of computations for encoding a codeword is fixed. As the number of codewords in a codebook is also fixed, let it be N. Hence, the additional time complexity of the codebook compression is $O(N)$. Therefore, the time complexity of the VQ algorithm is increased $O(N)$ if the codebook compression is integrated with the VQ.

5. Conclusions

In this study, the VQ algorithm is modified as a two-step compression method. This study method improves the compression ratio of the VQ algorithm. It reduces the size of the VQ codebook by 62.50%. The visual quality of encoded images using VQ and this method is almost similar in terms of PSNR and SSIM. In terms of storage space requirement, for images of size 256×256 and 512×512, it improves the overall performance of the VQ by up to 3%. Future work may be aimed at the compression of the index matrix to further improve the compression performance of VQ.

Conflicts of Interest

The authors declare no conflict of interest.

References

[1] A. Hasnat, D. Barman, S. Halder, and D. Bhattacharjee, "Modified Vector Quantization Algorithm to Overcome the Blocking Artefact Problem of Vector Quantization Algorithm," Journal of Intelligent & Fuzzy Systems, vol. 32, no. 5, pp. 3711-3727, 2017.

[2] R. Li, Z. Pan, and Y. Wang, "A General Codebook Design Method for Vector Quantization," Multimedia Tools and Applications, vol. 77, no. 18, pp.23803-23823, September 2018.

[3] L. Wang, Z. M. Lu, L. H. Ma, and Y. P. Feng, "VQ Codebook Design Using Modified K-Means Algorithm with Feature Classification and Grouping Based Initialization," Multimedia Tools and Applications, vol. 77, no. 7, pp. 8495-8510, April 2018.

[4] S. Yang and Y. Mao, "Vector Quantization of Deep Convolutional Neural Networks with Learned Codebook," 17th Canadian Workshop on Information Theory, pp. 39-44, June 2022.

[5] M. H. Vali and T. Bäckström, "NSVQ: Noise Substitution in Vector Quantization for Machine Learning," IEEE Access, vol. 10, pp. 13598-13610, January 2022.

[6] H. A. S. Leitao, W. T. A. Lopes, and F. Madeiro, "PSO Algorithm Applied to Codebook Design for Channel-Optimized Vector Quantization," IEEE Latin America Transactions, vol. 13, no. 4, pp.961-967, April 2015.

[7] A. Hasnatand D. Barman, "A Proposed Multi-Image Compression Technique," Journal of Intelligent & Fuzzy Systems, vol. 36, no. 4, pp.3177-3193, 2019.

[8] D. Barman, A. Hasnat, and B. Barman, "A Quantization Based Codebook Formation Method of Vector Quantization Algorithm to Improve the Compression Ratio While Preserving the Visual Quality of the Decompressed Image," Multidimensional Systems and Signal Processing, vol. 34, no. 1, pp. 127-145, March 2023.

[9] P. K. Shah, R.P. Pandey, and R. Kumar, "Vector Quantization with Codebook and Index Compression," International Conference System Modeling & Advancement in Research Trends, pp. 49-52, November 2016.

[10] J. Rahebi, "Vector Quantization Using Whale Optimization Algorithm for Digital Image Compression," Multimedia Tools and Applications, vol. 81, no. 14, pp. 20077-20103, June 2022.

[11] D. Barman, A. Hasnat, and B. Barman, "A Codebook Modification Method of Vector Quantization to Enhance Compression Ratio," High Performance Computing and Networking, vol. 853, pp. 227-234, 2022.

[12] D. Barman, A. Hasnat, and B. Barman, "An Enhanced Technique to Improve the Performance of Multi-Image Compression Technique," Advanced Computing and Intelligent Technologies, vol. 914, pp. 307-315, 2022.

[13] P. Chavan, B. Sheela Rani, M. Murugan, P. Chavan, and M. Kulkarni, "An Analysis of Codebook Optimization for Image Compression: Modified Genetic Algorithm and Particle Swarm Optimization Algorithm," Proceedings of Fourth International Conference on Communication, Computing and Electronics Systems, vol. 977, pp. 849-866, 2023.

[14] A. Hasnat, D. Barman, and B. Barman, "Luminance Approximated Vector Quantization Algorithm to Retain Better Image Quality of the Decompressed Image," Multimedia Tools and Applications, vol. 80, no. 8, pp.11985-12007, March 2021.

[15] U. Sara, M. Akter, and M. S. Uddin, "Image Quality Assessment Through FSIM, SSIM, MSE and PSNR—A Comparative Study," Journal of Computer and Communications, vol. 7, no. 3, pp. 8-18, March 2019.

[16] J. K. Mandal, Reversible Steganography and Authentication via Transform Encoding, Studies in Computational Intelligence, vol. 901, Singapore: Springer, 2020.