BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES, Vol. 73(5), 2025, Article number: e154734
DOI: 10.24425/bpasts.2025.154734

Lossless image compression method using vector quantization based on minimizing mean absolute error

Małgorzata FRYDRYCHOWICZ* **and Grzegorz ULACHA**

Faculty of Computer Science and Information Technology, West Pomeranian University of Technology, ul. Żołnierska 49, Szczecin, 71-210, Poland

Abstract. In this paper, we propose a novel lossless image compression method. During the prediction stage for each block of 8×8 pixels, a mechanism for preselecting one of N linear predictors from the dictionary is employed. The dictionary is determined individually for each encoded image using vector quantization (initially with a redundant number of vectors in the dictionary) and a fast algorithm that minimizes mean absolute error. In next steps, the prediction errors are encoded in a two-step manner using an adaptive Golomb code followed by an adaptive binary arithmetic coder. In this study, we demonstrate the efficiency of the proposed solution against other competitive codecs, including those based on deep learning. The proposed method offers high compression efficiency and is characterized by a short decoding time.

Keywords: entropy coding; lossless data compression; predictive models; vector quantization; iterative reweighted least squares.

1. INTRODUCTION

Cost optimization plays an important role in computer systems. Owing to data compression, costs can be lowered at both the transmission and data archiving levels. The memory requirements for storing multimedia data are especially challenging because of the high memory demand and transmission bandwidth. This paper focuses on lossless image compression, which finds its use in archiving various types of images such as medical 2D, 3D and 4D (three-dimensional video sequences) [1–5], astronomical or compressing satellite images [6]. Moreover, the lossless mode is often required during digital photo processing, creating advertising materials, film post-production, etc.

Compression methods typically consist of two stages: data decomposition to decrease information redundancy and data compression using one of efficient entropy coding methods, among which arithmetic and Huffman coding are the most efficient [7]. In the case of images, decorrelation can significantly reduce data redundancy due to the high level of dependency between adjacent pixels. At this stage, wavelet transforms are the most frequently used, e.g., JPEG2000 [8], SPIHT [9], ICER [10], as well as prediction methods JPEG-LS [11], CALIC [12]. In the majority of studies, linear or nonlinear prediction is used. Insight into different lossless coding approaches can be found in review papers [13–15].

The highest efficiency of lossless compression is achieved by algorithms with high computational complexity that belong to the time-symmetric class (where coding and decoding times are equally long). These solutions are based on linear prediction models with backward adaptation. They use mechanisms known from the literature, such as RLS [16], OLS [17–19], or WLS [20], where encoding and decoding of each subsequent pixel is accompanied by a procedure of adaptation or recalculation of linear predictor coefficients.

The latest methods are based on deep learning and use nonlinear neural networks [21–24]. They are also usually characterized by high computational complexity, and short (not in all cases) encoding/decoding time is achieved only because of the high level of parallelization, using GPU/NPU technologies [25–27]. Therefore, in this paper we propose a relatively efficient method that offers short decoding time without the need for dedicated high-performance computing units. This advantage is important because usually images are compressed once, whereas decoding is performed many times.

The basics of image modeling, which make it possible to compress data efficiently by reducing data redundancy, are discussed in Section 2. The original solution, which involves the initial construction of a dictionary containing N predictors individually calculated for each encoded image, is presented in Section 3. In Section 4, the bit average of the proposed solution and other popular codecs is compared.

2. BASICS OF IMAGE MODELING

One way to remove mutual information from encoded images is to use linear prediction with an appropriate selection of adjacent pixels and prediction order. Due to the direction of image encoding assumed in this paper (row by row, from top to bottom, starting from left to right), both the encoder and decoder have access to the pixels above and to the left of the currently encoded (decoded) pixel, which is described as the principle of causality. Using the assumption of decreasing correlation together with in-

*e-mail: frydrychowicz.malgorzata@zut.edu.pl

Manuscript submitted 2025-03-01, revised 2025-05-03, initially accepted for publication 2025-06-01, published in August 2025.

creasing distance between pixels, the neighbouring pixels of the currently encoded one can be numbered according to the increasing Euclidean distance $\sqrt{(\Delta x_j)^2+(\Delta y_j)^2}$ between them. The numbering of equally distant pixels is determined clockwise. This allows us to obtain a one-dimensional signal domain, which makes it easier to mathematically describe many equations and relations known from the literature regarding one-dimensional signals. Figure 1 illustrates the 48 nearest neighbouring pixels of the currently encoded pixel x_n, where the j-th index indicates a pixel of value $P(j)$. Theoretically, the higher the pixel index, the lower its impact on improving encoding efficiency.

		46	42	38	43	47			
	37	32	26	24	27	33	39		
36	29	20	16	14	17	21	30	40	
45	31	19	11	8	6	9	12	22	34	48
41	25	15	7	3	2	4	10	18	28	44
35	23	13	5	1	x_n					

Fig. 1. Neighbourhood pixel numbering

Various techniques are used in data modeling. However, in lossless image compression, a typical linear predictor of order r is most often used. In linear prediction, the predicted value of currently encoded pixel x_n is based on r neighbouring pixels (in accordance with the principle of causality known to the encoder and decoder). The linear predictor takes the following form

$$\hat{x}_n = \sum_{j=1}^{r} b_j \cdot P(j), \tag{1}$$

where elements $P(j)$ are the values of the nearest neighbouring pixels of the currently encoded pixel x_n, and b_j are the prediction coefficients forming a vector $\mathbf{B} = [b_1, b_2, ..., b_r]$ [7]. In practical solutions, it is often assumed, that the sum of the coefficients of such model should be equal to 1 (which is a condition for an unbiased prediction estimator). With this assumption and an 8-bit greyscale input values, the predicted value $\hat{x} \in \langle 0; 255 \rangle$. The use of linear or nonlinear predictor enables the encoding of prediction errors only, that is, the differences between the actual pixel values and predicted values (rounded to the nearest integer, because often the predicted values belong to the set of real numbers), which are usually small values oscillating near zero

$$e_n = x_n - [\hat{x}_n]. \tag{2}$$

In this way, we obtain a differential image in which the probability distribution of errors e_n is close to the geometric distribution, enabling efficient encoding of those errors using one of the entropy coding methods.

2.1. Predictive modeling methods with block division

By taking advantage of the variety of characteristics of different areas within a single image, it can be divided into blocks (e.g., 8×8 or 16×16 pixels). Each block is assigned an individual prediction model (in a form of r linear predictor coefficients in accordance with the formula (1)). One of the first solutions of this kind was the method presented in [28], where each 8×8 pixels block was assigned one static model from a dictionary consisting of 8 models in total (dictionary was predefined and constant), which produced the lowest mean absolute error. The header information associated with a single block required 3 bits, and with this data the prediction model index was identified (in general, each block is coded using one selected predictor from the dictionary of size N, meaning that the block is assigned a predictor from dictionary, and an index of this predictor must be saved in header data).

Further enhancements of this approach introduced the minimization of the mean square error (MMSE) as a method for determining the best set of prediction coefficients. However, the need to store a very large size of header information emerged, because of the large number of bits required to save prediction coefficients. In order to reduce size of header, blocks with similar characteristics were grouped together into clusters, with which a single shared prediction model was associated [29]. Using vector quantization techniques (as well as fuzzy clustering [30]), the optimized sets of, e.g., 16 prediction models were created. Owing to this, even with high prediction order the overall size of output file header did not significantly increase the bit average. In paper [31], a technique for combining adjacent blocks belonging to the same category (associated with the same predictor) into groups resulting in larger blocks was used. Then, a map of blocks of different sizes was saved using an efficient technique for coding quadtrees.

It is possible to obtain prediction coefficients that outperform predictors created using MMSE method in terms of lowering entropy [31]. In paper [32], the authors used minimum mean absolute error (MMAE), which allowed for better results compared to the use of MMSE for applications with block division. A wider explanation of the nonoptimal influence of MMSE on entropy minimization was presented in [33]. Therefore, in proposed solution, we decided to use a convenient method – iterative reweighted least squares (IRLS), to determine prediction coefficients based on minimizing mean absolute error criterion (see Section 3).

2.2. Cumulative prediction error correction method

In many cases, prediction methods can contain a constant component C_{mix}, whose value depends on the characteristics of the individual context. Hence, many solutions offer an adaptive method for C_{mix} removal (bias cancellation), also known as context-based error correction techniques, which improve the results of predictive modeling. In this case, the "context" is understood as a set of features resulting from the dependencies occurring between several nearest pixels of the currently encoded value x_n.

Adaptive context-dependent constant removal method is used i.e. in CALIC and JPEG-LS. For each context, the number of its occurrences M_i is recorded together with its accumulated sum of errors S_i. Based on these values, the currently determined prediction error is being corrected [12]. The value of constant component $C_{\text{mix}} = S_i / M_i$ is added to predicted value calculated

with main predictor, and after rounding the result to the nearest integer the final prediction error value is calculated as

$$e_n = x_n - [\hat{x}_n + C_{\text{mix}}]. \tag{3}$$

A broader description of our method for determining context number can be found in [34].

2.3. Components of proposed codec

The proposed solution is based on cascading approach (see Fig. 2), in which beside predicted value determined using linear prediction, a CDCCR (context-dependent constant component removal) block for removing constant component C_{mix} associated with certain context is used. The final blocks of the cascade are used for efficient prediction error e_n encoding using an adaptive Golomb coder and context-adaptive binary arithmetic coder (CABAC). The calculation of the constant component C_{mix} and our prediction error coding algorithm are described in detail in [34].

Algorithms 1 and 2 show the data processing steps in coder and decoder of the solution proposed in this paper, respectively. Before encoding with Algorithm 1, it is necessary to build a dictionary of N centroids based on vector quantization (described in Section 3) and determine the number of the best-fit centroid for each encoded square.

Algorithm 1. Encoder data processing steps

1: For each sequentially encoded pixel x_n:
2: Read predictor number (associated with a given 8×8 pixels square) from dictionary.
3: Determine predicted value (1) and prediction error e_n (3) after taking into account context-dependent constant component C_{mix}.
4: Convert prediction error e_n into a stream of bits using an adaptive Golomb coder.
5: Encode bitstream from step 4 using adaptive binary arithmetic coder.
6: If there are remaining pixels to encode, then return to step 2.

Algorithm 2. Decoder data processing steps

1: For each sequentially encoded pixel x_n:
2: Convert input bitstream using an adaptive binary arithmetic decoder, resulting in a Golomb codeword.
3: Convert the Golomb codeword into prediction error e_n form.
4: Read predictor number (associated with a given 8×8 pixels square) from dictionary.
5: Determine predicted value (1) and C_{mix}, and then add those values to e_n, to obtain decoded value of pixel x_n.
6: If there are remaining pixels to encode, then return to step 2.

3. LINEAR BLOCK PREDICTION METHOD

The compression method proposed in this paper utilizes a linear prediction, where each block of 8×8 pixels is associated with one predictor from dictionary (individually calculated for each encoded image (see Section 3.3)). In contrast to the classic k-means approach, we proposed a scheme with a redundant number of predictors in dictionary and a procedure for their reduction to the desired amount. The final set of these predictors is determined in several steps (initialization, vector quantization process).

The initialization process begins with calculating the individual predictor for each 8×8 pixels block by minimizing the mean absolute error (see Section 3.1). For example, an 512×512 px image will have 4096 individual predictors. In the next step this number has to be lowered to much smaller amount of $1.5N$ shared predictors, and this is done at the dictionary initialization stage (see Section 3.2). Classes group blocks with similar features and have one shared predictor (centroid), which is used to encode pixels of blocks belonging to certain class. After the initialization stage, blocks are reclassified (reassigned) into the class that best matches their characteristics. After this, the vector quantization procedure begins leading to the optimization of bit average. After each reclassification step, centroids are recalculated using blocks based on current assignment to classes. Thus, classes are adapting to the characteristics of regions encoded by them (to the set of blocks with similar characteristics). In contrary to the MRP codec [31] which uses MMSE, in this paper the MMAE (based on IRLS see Section 3.1) is used. Moreover, the initial number of classes is redundant and is later reduced to the specified N_{stop} level in the last stage of the predictor determination algorithm. To optimize compression efficiency the values: N and N_{stop} were selected experimentally depending on the resolution of compressed images (see Section 3.3).

3.1. Iterative reweighted least squares algorithm

In paper [32], it was proposed to use the minimum mean absolute error, which allowed (in the case of dividing the image into squares of size 8×8) to obtain better results compared to the use of MMSE. It is noteworthy that to determine the prediction model it is not necessary to know the optimal solution which guarantees mean absolute error minimization, but an approximate solution is sufficient [35]. To achieve this, instead of using improved simplex algorithms, a method that uses the classic MMSE to reduce the problem to the problem of minimizing weighted least squares (WLS) can be used. For this, an iterative approach, namely iterative reweighted least squares (IRLS) [36] with a parameter $p = 1$ is used, which enables to relatively quickly (compared to simplex method) obtain a suboptimal solution of mean absolute error minimization.

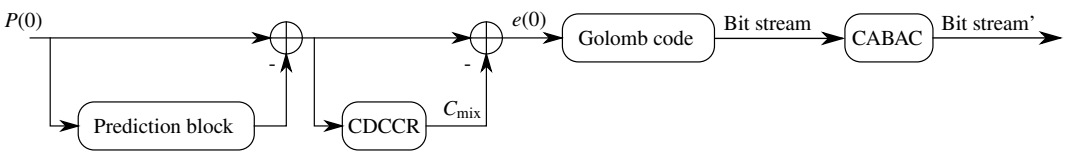

Fig. 2. Block diagram of cascade coding proposed in this paper

IRLS allows to get approximated l_p – norm solution minimizing average error for data in area Q

$$\|e\|_p = \left(\sum_{n \in Q} |e_n|^p \right)^{1/p}, \tag{4}$$

by solving the problem of minimization of the weighted least-squares, which offers relatively low computational complexity

$$\|e\|_p = \left(\sum_{n \in Q} w_n^2 \cdot |e_n|^2 \right)^{1/p}. \tag{5}$$

It is possible to perform an approximate minimization for any l_p – norm using an iterative algorithm that minimizes the expression

$$\|e\|_p = \left(\sum_{n \in Q} |e_n|^{(p-2)} \cdot |e_n|^2 \right)^{1/p}. \tag{6}$$

In the first iteration, weights are set to $w_n = 1$, and weighted least-squares minimization is performed using (5). Then, the equation is solved using Cholesky decomposition

$$\mathbf{B} = \mathbf{R}^{-1} \cdot \mathbf{P}, \tag{7}$$

where $\mathbf{R}$ is a $r \times r$ square matrix of elements $\mathbf{R}(j,i)$

$$\mathbf{R}(j,i) = \sum_{n \in Q} w_n^2 \cdot y_n(i) \cdot y_n(j), \tag{8}$$

where $j = \{1,2,\ldots,r\}$, $i = \{1,2,\ldots,r\}$, and $\mathbf{P}$ is a $r \times 1$ vector of $\mathbf{P}(j)$ elements

$$\mathbf{P}(j) = \sum_{n \in Q} w_n^2 \cdot x_n \cdot y_n(i), \tag{9}$$

where x_n denotes the value of the n-th sequentially encoded pixel and vector $\mathbf{Y}_n = [y_n(1), y_n(2), \ldots, y_n(r)]$. In proposed solution, the prediction model uses the r nearest neighbouring pixels (see Fig. 1), therefore the vector $\mathbf{Y}_n = [P_n(1), P_n(2), \ldots, P_n(r)]$. In next iterations of the IRLS algorithm, the weights are set using errors e_n^{old} received based on linear prediction model received in previous iteration using

$$w_n = \left| e_n^{\text{old}} \right|^{(p-2)/2}. \tag{10}$$

Each subsequent iteration in the IRLS algorithm converges towards the expected minimization accuracy. In the case of 8-bit data used in images, good results are achieved after approximately eight iterations. To minimize mean absolute error, a parameter $p = 1$. Then, substituting (10) into (5), we obtain the classical WLS problem

$$\|e\|_1 = \sum_{n \in Q} \frac{1}{|e_n^{\text{old}}|} \cdot |e_n|^2 \approx \sum_{n \in Q} |e_n|. \tag{11}$$

Special attention should be paid to the prediction errors oscillating around zero and preventing division by 0. Guard $|e_n^{\text{old}}| \leftarrow \max\{0.6, |e_n^{\text{old}}|\}$ gives good results.

3.2. Dictionary initialization

In order to speed up the convergence of the k-means algorithm, we proposed our own algorithm for initializing centroids dictionary (as opposed to the classical k-means approach, where initialization is done by random selection of N input data vectors). During the initialization process, a dictionary containing $1.5N$ classes that group blocks (8×8 pixels squares) with similar features are created. Later, during encoding stage, these blocks use a shared linear predictor (centroid) of the class to which they are assigned. As a part of the steps of the dictionary building algorithm, predictors are adapting to the characteristics of image, and the number of classes is gradually reduced to N_{stop}. The number of classes depends on the dimensions of the image – the larger the image, the more classes are used.

Prediction errors are encoded using a two-stage Golomb-CABAC coder (see Fig. 2). Due to the fact, that this coder adapts the probability distributions after each sequentially encoded value of $|e_n|$, it is problematic to minimize the bit average at the stage of determining the predictors assigned to classes (centroids). Therefore, a certain simplification can be made by calculating the bit average as follows

$$L_{\text{avg}} = H(S) + \frac{N_{\text{stop}} \cdot (r-1) \cdot (n_b + 2)}{\text{img height} \cdot \text{img width}} + \frac{\log_2 N_{\text{stop}}}{(\text{square size})^2}, \tag{12}$$

where n_b is the number of bits reserved for the fractional part of prediction coefficient, r is a prediction order, and $(r-1)$ is the number of coefficients saved into file (the first coefficient b_1 is skipped, because the sum of all coefficients b_j is equal to 1, therefore skipped coefficient can be easily reconstructed on decoder side), $H(S)$ is an entropy of prediction errors calculated using the following formula

$$H(S) = -\sum_{i=1}^{|e_{\max}|} p_i \cdot \log_2 p_i, \tag{13}$$

where p_i is the probability of occurrence of prediction error equal to i.

To determine the N_{stop} predictors offering the lowest possible bit average, a vector quantization algorithm must be used. Unlike classical methods such as LBG (k-means), in our case, the target function is defined differently and is based on the Minkowski distance. It is not a mean-square error but a function based on minimizing the mean prediction error values (in a given square)

$$d_i = \frac{1}{z} \sum_{j=1}^{z} |e_j|^{\alpha}, \tag{14}$$

where z is the number of pixels in a square, and i denotes square index. Depending on the phase of the algorithm, different values of parameter α are used.

The starting set of $1.5N$ classes is initialized in a hybrid manner: first $0.5N$ via Algorithm 3, and the remaining N via Algorithm 4.

Algorithm 3. First algorithm to determine class membership

1: For each square in the image, an individual linear predictor is calculated using the MMAE method, and each square is then encoded using (1) and (2).
2: For each square, a value of target function d_i with $\alpha = 1$ is calculated (14).
3: Results are sorted by the values of d_i assigned to each square and divided into $0.5N$ equal parts, creating a set of $0.5N$ initial classes.

Algorithm 4. Second algorithm to determine class membership

1: $s_i \leftarrow 0$
2: **for** $j \leftarrow 0, k-2$ **do**
3: **if** $b_j \geq \overline{b}_j$ **then**
4: $s_i \leftarrow s_i + 2^j$
5: **if** $\Delta_i > \overline{\Delta}$ **then**
6: $s_i \leftarrow s_i + 2^{k-1}$

The second method for determining centroids (Algorithm 4) is to calculate the arithmetic mean from all individual predictors at the beginning, creating an averaged vector $\overline{\mathbf{B}} = (\overline{b}_1, \ldots, \overline{b}_r)$. The class membership is determined using Algorithm 4.

Symbols in Algorithm 4 denotes:
i – square index,
s_i – class index assigned to the square of index i,
$N = 2^k$ – number of centroids,
and value of Δ_i and $\overline{\Delta}$ are calculated as

$$\Delta_i = \sum_{j=1}^{r} \overline{d}_j \cdot |b_j^{(i)} - \overline{b}_j|^{\gamma}, \tag{15}$$

$$\overline{\Delta} = \frac{1}{\text{number of squares}} \cdot \sum_{i=1}^{\text{number of squares}} \Delta_i \tag{16}$$

where $\gamma = 1.9$ (value chosen experimentally) and

$$\overline{d}_j = \frac{1}{\sqrt{(\Delta x_j)^2 + (\Delta y_j)^2}}, \tag{17}$$

where Δx_j and Δy_j represents horizontal and vertical distance between pixel x_n and $P(j)$ on Fig. 1.

Both sets created as a result of Algorithms 3 and 4 are then combined into a single set of size $1.5N$. Then, the assignments of squares to classes are removed. This is due to the fact, that by using two independent initialization algorithms working on the same set of squares, the assignments of squares to classes are created within each of these algorithms, meaning that the final assignment of squares will be doubled after combining sets into a single merged set. Therefore, after calculating predictors of initial set of $1.5N$ classes, the assignments are cleared, and squares are once again assigned to the nearest class at the reclassification stage (described in Section 3.3).

3.3. Building a dictionary of predictors

Initialization is the first stage of the dictionary building algorithm. It begins with the calculation of individual predictor for each block (8×8 pixels square) using the MMAE method. To achieve this, an iterative IRLS algorithm is used (described in Section 3.1), which offers fast calculation of the approximated solution. The number of iterations of the IRLS algorithm for calculating the individual predictors was set to 10.

After initialization (Section 3.2), having already created the first set of $1.5N$ classes, the quantization procedure (summarized in Algorithm 5) begins. Coder for t_1 iterations reclassifies blocks between classes (second stage of dictionary building). During reclassification, all blocks are encoded with each centroid (predictor associated with a single class). After block encoding, the target function (14) with parameter $\alpha = 1.2$ is calculated, which determines the measure of the block proximity to given class. Blocks are assigned to the class which offers the lowest values of d_i. At the end of each iteration of reclassification, the matrices $\mathbf{R}$ and vectors $\mathbf{P}$ of all blocks assigned to certain classes are added and using (7) based on MMAE (3 iterations of IRLS) a new shared predictor is calculated.

Algorithm 5. Quantization algorithm

1: Reclassification of squares to the closest class (in terms of criterion (14) with $\alpha = 1.2$).
2: Recalculation of centroids based on current squares assignment using MMAE (3 iterations of IRLS).
3: Repeat steps 1–2 for t_1 times.
4: Removal of the least used class and new assignment of its squares to the closest class (in terms of criterion (14) with $\alpha = 1.2$).
5: Recalculation of centroids based on current squares assignment using MMAE (3 iterations of IRLS).
6: Reclassification of squares to the closest class (in terms of criterion (14) with $\alpha = 1.2$).
7: Repeat steps 4–6 for $t_2 = 1.5N - N_{\text{stop}}$ times.
8: Reclassification of squares to the closest class (in terms of criterion (14) with $\alpha = 1.2$).
9: Recalculation of centroids based on current squares assignment using MMAE (3 iterations of IRLS).
10: Repeat steps 8–9 for t_3 times.

After completing t_1 iterations of the second stage of the algorithm, the class settings (centroid and assigned blocks) giving (within t_1 iterations) the lowest bit average (12) are saved and passed on to the third stage, where during $t_2 = 1.5N - N_{\text{stop}}$ iterations a number of $1.5N$ classes is gradually reduced to N_{stop} classes. In each iteration, the class assigned with the fewest blocks is removed and its blocks are assigned to the closest (in terms of criterion (14) with $\alpha = 1.2$) of the remaining classes. After this operation, the predictors of each class are determined again and all squares are reclassified. After reaching N_{stop} classes (where $N_{\text{stop}} \leq 1.5N$), the fourth stage takes place, during which an additional t_3 iterations of reclassification of all squares is performed (similarly to second stage).

Based on the experiments performed on 45 test images [42], consisting of images with three different resolutions: 256×256, 512×512, and 720×576 pixels, the individual parameters pre-

Table 1
Bit average based on a database of 45 standard test images for different codecs

Images	Pngcrush [37]	WebP [38]	WebP2 [39]	JPEG-XL [40]	MRP [41]	MRP optimized	Proposed codec
Average	4.719	4.216	4.266	4.123	4.058	4.022	3.983

Table 2
Bit average based on a test image dataset from paper [21]

Images	BPG	PNG	LCIC	JPEG 2000	JPEG-LS	JPEG-XL	FLIF	WebP	WebP2	L3C	CWPLIC	LCIC duplex	Proposed codec
Airplane	4.32	4.26	3.99	4.00	3.80	3.71	3.82	3.87	3.84	4.56	3.69	3.69	3.63
Barbara	5.06	5.22	4.61	4.61	4.70	4.40	4.56	4.55	4.51	5.44	4.35	4.36	3.94
Coastguard	5.70	5.06	4.82	4.83	4.86	4.73	4.93	4.81	4.82	5.82	4.80	4.83	4.41
Comic	6.15	5.84	5.63	5.65	5.30	5.07	5.50	5.45	5.39	6.60	4.83	4.83	5.01
Flowers	5.18	5.08	4.91	4.92	4.62	4.51	4.74	4.76	4.70	5.53	4.41	4.35	4.38
Goldhill	4.95	4.70	4.58	4.59	4.43	4.37	4.50	4.47	4.41	5.27	4.33	4.33	4.22
Lennagrey	4.54	4.61	4.31	4.31	4.24	4.16	4.28	4.14	4.13	4.95	4.13	4.08	3.96
Mandrill	6.61	6.23	6.11	6.11	6.04	5.98	6.14	5.89	5.90	6.97	5.95	5.89	5.74
Monarch	4.10	4.26	3.82	3.82	3.70	3.54	3.68	3.73	3.72	4.37	3.40	3.45	3.42
Pepper	4.77	4.90	4.63	4.63	4.51	4.48	4.58	4.50	4.47	5.38	4.67	4.38	4.28
Ppt3	2.20	2.35	2.41	2.41	2.04	1.84	1.87	2.06	2.01	3.71	2.14	2.07	1.93
Zebra	5.83	5.19	4.89	4.89	4.81	4.66	4.84	4.86	4.84	6.08	4.65	4.68	4.36
Average	4.951	4.808	4.559	4.564	4.421	4.288	4.453	4.424	4.395	5.390	4.279	4.245	4.113

sented in Table 3 were selected. Larger images, which may contain many regions with different characteristics are encoded using a larger number of classes in contrast to smaller images. The prediction order r is approximately 35. The number of iterations $t_1 = t_3 = 20$ was also selected.

In practice, nearly 98% of the coded data are compressed prediction errors. About 1.5% is the cost of storing indexes (to the rows of dictionary) assigned to each square. The size of the dictionary, on the other hand, is only about 0.5% of the size of the encoded file. It has N_{stop} rows, and each row consists of r prediction coefficients, which are (n_b+2)-bit (see Table 3). The size of dictionary equals $N_{\text{stop}} \cdot (n_b+2) \cdot (r-1)$ bits, where N_{stop} is a number of classes (shared predictors), n_b+2 is a number of bits required to save single prediction coefficient, and r is a prediction order (see (1)).

Table 3
Coding parameters for images with different resolutions

Resolution	r	N	N_{stop}	n_b
$\leq 256\times256$	35	4	6	9
$\leq 512\times512$	36	16	16	10
$> 512\times512$	35	32	32	10

Table 4
Time statistics for MRP codec and proposed codec

	MRP optimized	Proposed codec
Encoding time [s]	565.584	263.837
Decoding time [s]	0.062	0.327

4. EXPERIMENTS

The bit averages of several known lossless compression methods are compared in Table 1. Experiments were performed on images with different features (with small and large variations of noise, photographs, artificially generated images with soft tonal gradations, images with textured areas etc.). The configurations of tested codecs were set to achieve the best results and is presented in Table 5.

From [21], we decided to use another set of test images (in 8-bit grayscale) (see Table 2), because it is one of the few com-

Table 5
Configuration of tested codecs

Codec	Configuration
Pngcrush	-reduce -brute
WebP	-lossless -m 6 -q 100
WebP2	-q 100 -alpha_q 100 -effort 9
JPEG-XL	–distance=0 –effort=9
MRP	with default configuration
MRP optimized	-o

parisons referring to the use of neural networks (LCIC, L3C, CWPLIC, LCIC duplex codecs), in which bit averages for individual encoded images are shown (unfortunately, in the case of many deep learning papers, there is usually one average result for the entire image database, which makes it difficult to analyze in more depth the capabilities offered by individual codecs for different types of images). The authors of mentioned paper also included several classical codecs (e.g., accepted by JPEG [43]) in their comparison.

In Table 4 the time statistics are shown for coding an example image Lennagrey (512×512 pixel) using processor AMD Ryzen 7 5700x 3.4GHz. The statistics were collected for two solutions based on linear prediction with block division. The results presented in Tables 1 and 4 show the advantage of the proposed solution over MRP Optimized in terms of both bit average and encoding time.

5. CONCLUSIONS

In this paper, a method for lossless image compression was presented. At the prediction stage, for each block of 8×8 pixels, a mechanism of preselecting one of the N linear predictors from the dictionary was employed. The dictionary was calculated individually for each encoded image using original vector quantization method and fast mean absolute error minimization. After calculating prediction error, a simple method for correcting the cumulative prediction error was used. Such prepared prediction errors were coded in a two-step manner using the Golomb code followed by an adaptive binary arithmetic coder.

The developed method is asymmetric in terms of time, and offers a relatively short decoding time. For example, for the image Lennagrey (512×512 pixels) (using AMD Ryzen 7 5700x 3.4GHz processor) the encoding time is shorter by 53% comparing to MRP – Optimized codec and is equal to 263.837 s (see Table 4). Although there are methods offering compression efficiencies higher by a few percent, the decoding time is usually higher even several times (e.g., Vanilc WLS-D [44], Blend-28 [34]).

For the future development of proposed codec, it is planned to improve compression efficiency by introducing compression of the prediction coefficients dictionary, and improving the mechanism of dictionary initialization. The introduction of fuzzy quantization at the building stage of the dictionary is also considered. Importantly, this will not affect the decoding time compared to the solution presented in this paper.

REFERENCES

[1] A. Kassim, P. Yan, W.S. Lee, and K. Sengupta, "Motion compensated lossy-to-lossless compression of 4-D medical images using integer wavelet transforms," *IEEE Trans. Inf. Technol. Biomed.*, vol. 9, no. 1, pp. 132–138, Mar. 2005, doi: 10.1109/TITB.2004.838376.

[2] V. Sanchez, P. Nasiopoulos, and R. Abugharbieh, "Efficient 4D motion compensated lossless compression of dynamic volumetric medical image data," in *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, Apr. 2008, pp. 549–552, doi: 10.1109/ICASSP.2008.4517668.

[3] J. Scharcanski, "Lossless and near-lossless compression for mammographic digital images," in *2006 International Conference on Image Processing*, Atlanta, GA, USA, Oct. 2006, pp. 2253–2256, doi: 10.1109/ICIP.2006.312811.

[4] J. Ström and P.C. Cosman, "Medical image compression with lossless regions of interest," *Signal Process.*, vol. 59, no. 2, pp. 155–171, Jun. 1997, doi: 10.1016/S0165-1684(97)00044-3.

[5] X. Xie, L. GuoLin, and W. ZhiHua, "A Near-Lossless Image Compression Algorithm Suitable for Hardware Design in Wireless Endoscopy System," *EURASIP J. Adv. Signal Process.*, vol. 2007, p. 082160, Dec. 2007, doi: 10.1155/2007/82160.

[6] X. Chen, C. Canagarajah, R. Vitulli, and J. Nunez-Yanez, "Lossless Compression for Space Imagery in a Dynamically Reconfigurable Architecture," in *Proceedings of International Workshop on Applied Reconfigurable Computing (ARC2008)*, vol. 4943, Mar. 2008, pp. 336–341, doi: 10.1007/978-3-540-78610-8_38.

[7] K. Sayood, *Introduction to Data Compression*, 5th ed. San Francisco: Morgan Kaufmann, 2018, doi: 10.1016/C2015-0-06248-7.

[8] M. Marcellin, M. Gormish, A. Bilgin, and M. Boliek, "An overview of JPEG-2000," in *Proceedings DCC 2000. Data Compression Conference*, Snowbird, UT, USA, Mar. 2000, pp. 523–541, doi: 10.1109/DCC.2000.838192.

[9] A. Said and W. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 3, pp. 243–250, Jun. 1996, doi: 10.1109/76.499834.

[10] A. Kiely and M. Klimesh, "The ICER Progressive Wavelet Image Compressor," *Interplanet Network Progress Report 42*, vol. 155, pp. 1–46, Nov. 2003.

[11] M. Weinberger, G. Seroussi, and G. Sapiro, "The LOCO-I lossless image compression algorithm: principles and standardization into JPEG-LS," *IEEE Trans. Image Process.*, vol. 9, no. 8, pp. 1309–1324, Aug. 2000, doi: 10.1109/83.855427.

[12] X. Wu and N. Memon, "CALIC-a context based adaptive lossless image codec," in *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, vol. 4, Atlanta, GA, USA, May 1996, pp. 1890–1893 vol. 4, doi: 10.1109/ICASSP.1996.544819.

[13] B. Carpentieri, M. Weinberger, and G. Seroussi, "Lossless compression of continuous-tone images," *Proc. IEEE*, vol. 88, no. 11, pp. 1797–1809, Nov. 2000, doi: 10.1109/5.892715.

[14] G. Deng, "Transform domain LMS-based adaptive prediction for lossless image coding," *Signal Process-Image Commun.*, vol. 17, no. 2, pp. 219–229, Feb. 2002, doi: 10.1016/S0923-5965(01)00019-4.

[15] N. D. Memon and K. Sayood, "Lossless image compression: a comparative study," in *Still-Image Compression*, vol. 2418. SPIE, Mar. 1995, pp. 8–20, doi: 10.1117/12.204127.

[16] G. Ulacha and R. Stasinski, "Context based lossless coder based on RLS predictor adaption scheme," in *2009 16th IEEE International Conference on Image Processing (ICIP)*, Egypt, Cairo, Nov. 2009, pp. 1917–1920, doi: 10.1109/ICIP.2009.5413680.

[17] X. Wu, E. Barthel, and W. Zhang, "Piecewise 2D autoregression for predictive image coding," in *Proceedings 1998 International Conference on Image Processing. ICIP'98*, vol. 3, Chicago, Illinois, USA, Oct. 1998, pp. 901–904, doi: 10.1109/ICIP.1998.727397.

[18] H. Ye, G. Deng, and J. Devlin, "Adaptive linear prediction for lossless coding of greyscale images," in *Proceedings 2000 International Conference on Image Processing (Cat. No.00CH37101)*, vol. 1, Vancouver, BC, Canada, Sep. 2000, pp. 128–131 vol.1, doi: 10.1109/ICIP.2000.900911.

[19] H. Ye, G. Deng, and J. Devlin, "Least squares approach for lossless image coding," in *ISSPA '99. Proceedings of the Fifth International Symposium on Signal Processing and its Applications*, vol. 1, Brisbane, Queensland, Australia, Aug. 1999, pp. 63–66, doi: 10.1109/ISSPA.1999.818113.

[20] G. Ulacha, R. Stasiński, and C. Wernik, "Extended Multi WLS Method for Lossless Image Coding," *Entropy*, vol. 22, no. 9, Aug. 2020, doi: 10.3390/e22090919.

[21] H. Rhee, Y.I. Jang, S. Kim, and N.I. Cho, "Lossless Image Compression by Joint Prediction of Pixel and Context Using Duplex Neural Networks," *IEEE Access*, vol. 9, pp. 86 632–86 645, Jun. 2021, doi: 10.1109/ACCESS.2021.3088936.

[22] Z. Zhang, H. Wang, Z. Chen, and S. Liu, "Learned Lossless Image Compression Based on Bit Plane Slicing," in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA, Jun. 2024, pp. 27 569–27 578, doi: 10.1109/CVPR52733.2024.02604.

[23] H. Rhee and N. Cho, "Resolution-Adaptive Lossless Image Compression Using Frequency Decomposition Network," Taipei, Taiwan, Oct. 2023, pp. 688–695, doi: 10.1109/APSIPAASC58517.2023.10317322.

[24] X. Feng, E. Gu, Y. Zhang, and A. Li, "Probability Prediction Network With Checkerboard Prior for Lossless Remote Sensing Image Compression," *IEEE J. Sel. Top. Appl. Earth Observ. Remote Sens.*, vol. 17, pp. 17 971–17 982, 2024, doi: 10.1109/JSTARS.2024.3462948.

[25] Y. Shen *et al.*, "GPU-accelerated Lossless Image Compression with Massive Parallelization," in *2023 IEEE International Symposium on Multimedia (ISM)*, Laguna Hills, CA, USA, Dec. 2023, pp. 321–324, doi: 10.1109/ISM59092.2023.00061.

[26] A. Said, H. Le, and F. Farhadzadeh, "Bitstream Organization for Parallel Entropy Coding on Neural Network-based Video Codecs," in *2023 IEEE International Symposium on Multimedia (ISM)*. Los Alamitos, CA, USA: IEEE Computer Society, Dec. 2023, pp. 1–9, doi: 10.1109/ISM59092.2023.00007.

[27] Y. Bai, X. Liu, K. Wang, X. Ji, X. Wu, and W. Gao, "Deep Lossy Plus Residual Coding for Lossless and Near-Lossless Image Compression," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 5, pp. 3577–3594, May 2024, doi: 10.1109/TPAMI.2023.3348486.

[28] N. Memon and K. Sayood, "An asymmetric lossless image compression technique," in *Proceedings of the 1995 International Conference on Image Processing*, vol. 3, Washington, DC, USA, Oct. 1995, pp. 97–100, doi: 10.1109/ICIP.1995.537589.

[29] F. Golchin and K. Paliwal, "Classified adaptive prediction and entropy coding for lossless coding of images," in *Proceedings of International Conference on Image Processing*, vol. 3, Santa Barbara, CA, USA, Oct. 1997, pp. 110–113, doi: 10.1109/ICIP.1997.632006.

[30] B. Aiazzi, L. Alparone, and S. Baronti, "Near-lossless image compression by relaxation-labelled prediction," *Signal Process.*, vol. 82, no. 11, pp. 1619–1631, Nov. 2002, doi: 10.1016/S0165-1684(02)00305-5.

[31] I. Matsuda, N. Ozaki, Y. Umezu, and S. Itoh, "Lossless coding using variable block-size adaptive prediction optimized for each image," in *2005 13th European Signal Processing Conference*, Sep. 2005, pp. 1–4.

[32] Y. Hashidume and Y. Morikawa, "Lossless image coding based on minimum mean absolute error predictors," in *SICE Annual Conference 2007*, Takamatsu, Japan, Sep. 2007, pp. 2832–2836, doi: 10.1109/SICE.2007.4421471.

[33] X. Wang and X. Wu, "On Design of Linear Minimum-Entropy Predictor," in *2007 IEEE 9th Workshop on Multimedia Signal Processing*, Chania, Greece, Oct. 2007, pp. 199–202, doi: 10.1109/MMSP.2007.4412852.

[34] M. Frydrychowicz and G. Ulacha, "Two-Stage Golomb – Context-Adaptive Binary Arithmetic Coders Coding in Lossless Image Compression," *Adv. Sci. Technol. Res. J.*, vol. 18, no. 6, pp. 62–85, Aug. 2024, doi: 10.12913/22998624/191111.

[35] G. Ulacha and M. Łazoryszczak, "Lossless Image Coding Using Non-MMSE Algorithms to Calculate Linear Prediction Coefficients," *Entropy*, vol. 25, no. 1, pp. 1–19, Jan. 2023, doi: 10.3390/e25010156.

[36] S. Burrus, "Iterative Reweighted Least Squares * C. OpenStax-CNX module: m45285," 2018.

[37] "Codec Pngcrush 1.8.11," 2017. [Online]. Available: https://sourceforge.net/projects/pmt/files/pngcrush-executables/1.8.11/ (Accessed 2024-09-12).

[38] "Codec WebP 1.3.2," 2023. [Online]. Available: https://storage.googleapis.com/downloads.webmproject.org/releases/webp/libwebp-1.3.2-windows-x64.zip (Accessed 2024-09-12).

[39] "Codec WebP2," 2025. [Online]. Available: https://chromium.google source.com/codecs/libwebp2/+/b0e76b0ef506021c7d39668e1b8ae7cb626e6ee3 (Accessed 2024-09-12).

[40] "Codec JPEG-XL 0.9.1," 2024. [Online]. Available: https://github.com/libjxl/libjxl/releases/download/v0.9.1/jxl-x86-windows-static.zip (Accessed 2024-09-12).

[41] "Codec MRP 0.5," 2005. [Online]. Available: https://www.rs.tus.ac.jp/matsuda-lab/matsuda/mrp/mrp-05.tar.gz (Accessed 2024-09-12).

[42] "Dataset of 45 Images," 2022. [Online]. Available: https://kakit.zut.edu.pl/fileadmin/Test_Images.zip (Accessed 2024-09-12).

[43] Y. Liang *et al.*, "Review of Static Image Compression Algorithms," in *2024 IEEE 7th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, vol. 7, Chongqing, China, Mar. 2024, pp. 222–231, doi: 10.1109/IAEAC59436.2024.10503622.

[44] A. Weinlich, P. Amon, A. Hutter, and A. Kaup, "Probability Distribution Estimation for Autoregressive Pixel-Predictive Image Coding," *IEEE Trans. Image Process.*, vol. 25, no. 3, pp. 1382–1395, Mar. 2016, doi: 10.1109/TIP.2016.2522339.