Advances in Science and Technology Research Journal

Advances in Science and Technology Research Journal, 2026, 20(4), 518–537
https://doi.org/10.12913/22998624/215213
ISSN 2299-8624, License CC-BY 4.0

Received: 2025.11.02
Accepted: 2026.02.15
Published: 2026.03.01

Initialization algorithm for lossless image compression with vector quantization

Małgorzata Frydrychowicz[1], Grzegorz Ulacha[1*]

[1] Faculty of Computer Science and Information Technology, West Pomeranian University of Technolgy in Szczecin, ul. Żołnierska 49, 71-210, Szczecin, Poland
* Corresponding author's e-mail: grzegorz.ulacha@zut.edu.pl

ABSTRACT

In this article we propose a novel algorithm for lossless image compression based on 8 × 8 px block division and vector quantization. Each block is assigned to one of k classes and encoded using linear predictor assigned to a class (calculated using the Iterative Reweighted Least Squares (IRLS) algorithm). Several methods for initializing classes based on binary division of a set are proposed. Their advantages and disadvantages are shown, and the best performing ones are selected in order to develop an original dictionary initialization algorithm, which is used in the vector quantization method adapted to the lossless image compression. The hierarchical, binary tree-based initialization method is a combination of these algorithms, in which class initialization procedure follows the pattern of a complete binary tree with k leaves. The proposed initialization method significantly reduced time required for the main vector quantization process. Proposed codec belongs to the time-asymmetric compression methods with a short decoding time and is characterized by high compression efficiency, offering on average a 7.22% lower bit average compared to JPEG-LS.

Keywords: lossless image compression, vector quantization, image block division, binary tree.

INTRODUCTION

Every year, more and more data is being generated worldwide. Its rapid growth is driven by both the development of technology and by widespread access to it. Data can be represented in the form of text documents, audio files, videos, image files, etc., making data compression applicable in a wide variety of areas. Despite decreasing cost of memory, there is still a need to efficiently store data and transfer it between devices while minimizing bandwidth consumption. Data compression methods allow to reduce the size of a message and are divided into two main categories: lossy and lossless compression [1, 2]. Lossy compression enables higher compression rates, but it involves irreversible loss of some data. It finds its use where the loss of less important elements from a perception point of view is acceptable. Lossless compression can be found in the archiving of medical images [3, 4] or satellite imagery compression [5]. In these cases, partial loss of data can cause errors in interpretation, conclusions, and decisions. There is also a third category of compression methods, namely, near-lossless compression, in which users are allowed to define a level of accepted data loss [6].

Depending on the criteria, lossless image compression methods can be divided into different categories. Taking the redundancy removal method as an example, in most cases, linear and nonlinear predictions are used, and the pixels are encoded sequentially, row by row, from left to right [7]. Other approaches include combining lossy prediction with lossless compression of prediction errors (an idea known from [8]). In [9], it was proposed to combine lossy BPG encoding (as a mechanism for determining approximate pixel values), which has a high degree of compression, with an additional stage of lossless compression of prediction errors. Meanwhile, in [10] a division into subimages (with lower resolution),

where one subset is encoded using JPEG-XL, and then remaining subimages are encoded through deep neural networks (LC-FDNet).

Implementation complexity is another criterion used to classify compression methods. There are two main categories to consider: time-symmetric and time-asymmetric. The first group of codecs is characterized by a short time of both encoding and decoding (when encoding an image with a resolution of 512 × 512 px it is usually counted in milliseconds). The codecs in this category are: JPEG-LS [11], CALIC [12] or more modern WebP [13].

The second group can also be included in the time-symmetric methods. However, they are characterized by long encoding and decoding time due to prediction methods using backward adaptation (codec parameters are tuned based on previously encoded/decoded data, and this adaptation usually occurs after each subsequently encoded pixel). The advantage of this solution is that there is no need to pass these parameters in the header section of the output file (which gives more flexibility e.g. in choosing the order of prediction), and the disadvantage is a low efficiency in the initial phase of encoding (due to small amount of learning data). This category includes methods such as OLS [14, 15], WLS [16], Vanilc WLS-D [17] and many methods that utilize neural networks.

In the first attempts to use neural networks for lossless image compression, an adaptive method was used, where the weights of network were modified on the fly after encoding each subsequent pixel (methods with backward adaptation). This is a real-time network learning without prior training using image training databases. In works [18, 19], an adaptive neural network (AdNN) was used, whose architecture is based on the multilayer perceptron layer (MLP), while in [20] a cellular neural network (CNN) was employed. Other, more recent approaches with deep learning have been discussed in more detail, among others, in the works [21, 22]. In [23] an comparison of several neural network solutions for image compression can be found, with a conclusion, that methods like PixelCNN (e.g. [24]) are generally computationally expensive. Moreover, the disadvantage of these methods is their strong dependence on the training datasets (which can easily suffer from overfitting, especially in case of very small images such as 32 × 32 or 64 × 64, resulting in insufficient efficiency for broader applications compared to conventional solutions), which was criticized in [25]. Confirmation of this uncomfortable conclusion can also be found in [26], where the authors explicitly stated that for high-resolution images, conventional codecs still have an advantage (offering a lower bit average) over learning-based methods. Another group of researchers, after a thorough analysis of the literature on deep neural networks (DNNs), also agreed that existing methods of lossless image encoding based on learning usually suffer from slow encoding speed. It is difficult to apply them to practical full-resolution image compression tasks, especially when one cannot rely on highly efficient multiprocessor GPU units.

Unlike methods with backward adaptation in conventional methods with forward adaptation, we can "tune" the prediction model more precisely owing to the complete knowledge of all image pixels. However, this introduces an additional cost of storing header data filled with information necessary for decoder to correctly decompress an image. Tuning parameters to the characteristics of given images usually requires many encodings, placing forward adaptation methods into a third category of codecs which are characterized by long encoding time and short decoding time (time asymmetric). The most efficient solutions in this group include, among others, MRP 0.5 [28] and 7-ctx MMAE [29].

Due to the constantly increasing performance of computers, implementations of algorithms with high complexity may be considered time acceptable in the upcoming years. An example is the use of vector quantization adapted for lossless image compression. Although this approach allows for high compression efficiency, it has been presented in only a small number of studies so far, among which [28, 30] is worth mentioning. In this approach, the initial stage of defining a dictionary consisting of k vectors is crucial. In the referenced works, the k-means algorithm was used, and the discussion of the initialization stage was either omitted or limited to a single sentence of commentary. For example, in [31], it was stated that the k-means algorithm is not very sensitive to the dictionary initialization method. In our opinion, this is not an accurate statement, especially regarding computation time (the number of k-means algorithm iterations). Therefore, in this study, a novel algorithm for a heuristic approach to dictionary initialization is proposed.

This study focuses on time-asymmetric methods (with forward adaptation) with fast decoding

mechanism, as the encoding operation is most often performed once, whereas the decoding operation is performed multiple times.

The basics of image modeling, which allows for efficient compression by redundancy removal, are presented in section Vector Quantization – From Lossy to Lossless Compression. Section Methods of Dictionary Initialization presents the basic principles of both lossy and lossless image compression based on vector quantization and a proposal of original initialization method that offers a significant reduction in the time required for main vector quantization process. In section Methods For Improving Compression Efficiency additional redundancy reduction methods are discussed, together with a comparison of the efficiency of the proposed solution against other popular codecs.

VECTOR QUANTIZATION – FROM LOSSY TO LOSSLESS COMPRESSION

Basics of lossless compression

Data modeling (also known as data decorrelation) is the process of removing as much mutual information as possible from an input stream. As a result, the entropy of the data decreases, and it can be compressed more efficiently. To achieve this, a pixel value can be predicted using linear prediction

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

where: r denotes prediction order, b_j is the prediction coefficient from vector $\mathbf{B} = [b_1, b_2, ..., b_r]$ and $P(j)$ is the value of the nearest j-th neighboring pixel of currently encoded pixel x_m.

In the case of images, to use this formula, a transformation of image into one-dimensional data is needed which is done by numbering the pixels of the neighborhood according to the increase in Euclidean distance $\sqrt{(\Delta x_j)^2 + (\Delta y_j)^2}$ between them. The numbering of equally distant pixels is determined clockwise. Pixel numbering scheme of the neighborhood of the currently encoded pixel x_m is shown in Figure 1. Theoretically, the farther away the pixel is from the pixel x_m, the smaller the relationship between the two (less impact on prediction).

The value of $\hat{x}_m$ rounded to the nearest integer is subtracted from the actual value of the pixel, giving a prediction error e_m, which is the subject of encoding

$$e_m = x_m - [\hat{x}_m] \tag{2}$$

For 8-bit data, the predicted values are in the range <0; 255>, prediction errors are usually small values close to zero and their probability distribution is approximately geometric, allowing for efficient compression.

Basics of lossy compression based on vector quantization

Vector quantization is commonly used in data processing, and the methods for constructing

			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_m					

Figure 1. Neighborhood pixel numbering

a dictionary are among NP-hard problems. In case of lossy image compression, the image is most commonly divided into square blocks with a side length of e.g. $B_S = 4$. To simplify, we will consider only luminance component and use grayscale images, where a pixel value is an 8-bit number between 0 and 255. In such case, each block of 4 × 4 px can be saved as a vector $\mathbf{x}_j = [x_{j,1}, x_{j,2}, ..., x_{j,16}]$ consisting of 16 elements (consecutive pixels read line by line). As a result, the entire image can be represented as a set of vectors $(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n)$, where n = height · width/B_S^2. For example, an image with a resolution 256 × 256 px and a block size of $B_S = 4$ will give $n = 4096$ vectors. These vectors can be quantized by introducing a dictionary of k centroids $(\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_k)$. To provide compression, the number of centroids should be significantly lower than n. Centroids $\mathbf{c}_i = [c_{i,1}, c_{i,2}, ..., c_{i,16}]$ are determined using, for example, the k-means algorithm, which aims to minimize the mean squared error (MSE). By design, centroid $\mathbf{c}_i$ represents the i-th class, that is, a certain set of vectors $\mathbf{x}_j$ with similar characteristics, which is why in the k-means algorithm $\mathbf{c}_i$ is calculated as the arithmetic mean of these vectors.

In lossy compression, the compression ratio is usually predefined, with the goal of achieving the lowest possible mean square error, the value of which is embedded in the PSNR formula used for the objective assessment of lossy compression quality (similarly to other metrics such as SSIM). Equation for 8-bit data is as follows:

$$PSNR = 10 \cdot \log_{10} \frac{255^2}{MSE} \tag{3}$$

In the simplest case of lossy compression, a fixed number of k centroids is set, making the degree of compression easy to define. In addition to the entire dictionary of k centroids, the output file contains a set of n indices to the dictionary. In place of each square represented by the vector $\mathbf{x}_j$, the decoder inserts the best-matched centroid $\mathbf{c}_i$. The cost of a single index is only $\log_2 k$, which is significantly less than the information about all the B_s^2 pixels of a given square $\mathbf{x}_j$.

The length of the encoded file, expressed in bits, can be calculated using the formula (4), which is the sum of two components. The first part is the size of the dictionary, and the second part is the cost of saving indices to the dictionary:

$$L_f = 8k \cdot B_S^2 + n \cdot \log_2 k \tag{4}$$

For example, for an image with a resolution of 256 × 256 px, block size $B_S = 4$ and dictionary consisting of $k = 256$ centroids, the file size is:

$L_f = 8 \cdot 256 \cdot 4^2 + \frac{256 \cdot 256}{4^2} \cdot \log_2 256 = 32768 = 2^{16}$ bits, which gives bit average $L_{avg} = L_f$/(height ·width) = 1 bit/pixel (compression ratio of 8:1). In this case, the size of dictionary takes 50% of the output file size. For images with higher resolutions, this aspect ratio will decrease unless the size of the dictionary itself is increased.

Lossless compression based on vector quantization

In order to switch from lossy to lossless compression, the output file besides L_f bit long header, must also include appropriately encoded errors. Errors are the differences between pixel values contained in the $\mathbf{x}_j$ vectors and their representatives which are part of the corresponding centroid $\mathbf{c}_i$. The number of errors equals the number of pixels in the image, and they shall be efficiently compressed using, for example, Huffman coding or arithmetic coding [6]. For such codes, the lower bound of the bit average L_{avg} is the entropy value $H(S)$ of the prediction errors, and entropy can be used as a helpful metric for the minimization of L_{avg}.

In practice, instead of using a set of fixed centroids (for a given image) obtained through k-means, in case of lossless compression, a linear prediction can be used. It enables calculation of the predicted value $\hat{x}_{j,m}$ for each encoded pixel $x_{j,m}$ based on r nearest neighboring pixels (see Figure 1). This implies that current centroid $\mathbf{c}_i = [c_{i,1}, c_{i,2}, ..., c_{i,16}]$ takes temporary form $\mathbf{c}_i = [\hat{x}_{i,1}, \hat{x}_{i,2}, \ldots, \hat{x}_{i,16}]$. Since predicted values are calculated on an ongoing basis, there is no need to store the dictionary in its current form (as in the case of lossy compression). However, following the principles of compression using vector quantization, we still deal with k classes. Each of the k classes is assigned an individual linear predictor $\mathbf{b}_i = [b_{i,1}, b_{i,2}, ..., b_{i,r}]$ of order r, and the values of vector $\mathbf{c}_i = [\hat{x}_{i,1}, \hat{x}_{i,2}, \ldots, \hat{x}_{i,16}]$ are calculated consecutively using (1). It is assumed that coefficients $b_{i,m}$ are in range <-1.999; 1.999>, and their sum equals 1. In the output file they are saved on $(n_b + 2)$ bits. Usually, an accuracy of $n_b = 9$ or 10 bits for the fractional part is sufficient. In addition, one bit is allocated for the sign of the number and one bit for the integer part. Experiments proved that, in contrast to lossy compression, for

lossless compression the preferable block size is 8 × 8 px (or larger), and the number of classes is significantly smaller and depends on the size of the image. For example, for images with a resolution of 512 × 512 px, a number of classes $k = 16$ works well. For this reason, the dictionary size of $k \cdot (r-1) \cdot (n_b + 2)$ bits is relatively small compared with the amount of data required to encode prediction errors. For example, when $k = 16$, $r = 36$, and $n_b = 10$, the dictionary consists of only 6720 bits. There are also fewer indices for the dictionary than in the case of lossy compression because of the significantly larger block size. Furthermore, with a considerably smaller number of centroids, the cost of saving each index equal to $\log_2 k$ is also smaller. For 512 × 512 px images the total cost required for indices (when $B_S = 8$, $k = 16$) equals $n \cdot \log_2 k = 4096 \cdot 4 = 16384$ bits. Therefore, the sum of two components making the header section is only $L_f = 23104$ bits, which translates to 0.08813 bit/pixel. It is approximately 2% of encoded data, whereas the remaining 98% is a cost of efficiently compressed prediction errors. The simplified formula for the bit average is as follows:

$$L_{avg} = H(S) + \frac{k \cdot (r-1) \cdot (n_b + 2)}{width \cdot height} + \frac{\log_2 k}{B_S^2} \quad (5)$$

The higher the prediction order and the larger the number of classes, the lower the entropy $H(S)$ becomes, but the header section size increases – parameter k affects the last two parts of formula (5). Hence, there is a need to search for compromise sets of three settings $\{k, r, n_b\}$ which will enable to simplify the process of minimizing the bit average by reducing it to entropy minimization at the vector quantization stage. Table 1 presents the experimentally chosen parameters that depend on the size of the image selected based on the training image dataset (distinct from the test images mentioned in section Compression efficiency). The Figure 2 presents relation between prediction order and bit average which was measured on a subset of images with resolution 512 × 512 px using proposed codec.

In addition, attention should also be paid to the problem of complexity of the k-means algorithm after it is adapted to lossless image compression. Although the procedure of checking the

Table 1. Coding parameters dependent on the image resolution

Resolution	r	n_b	k
Up to 256 × 256	36	9	6
Up to 512 × 512	36	10	16
Larger than 720 × 576	35	10	32

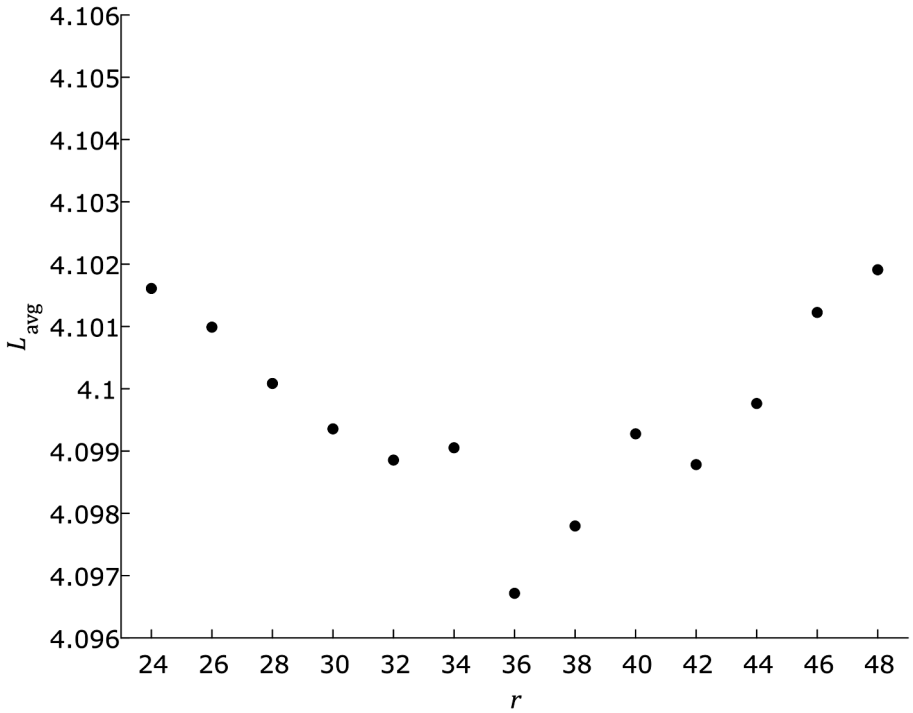

Figure 2. Results of bit averages for different values of prediction order r measured on images with resolution 512 × 512 px using proposed codec (VQ-MMAE$_2$)

best matching i-th centroid to the given vector $\mathbf{x}_j$ is linearly dependent on the number of pixels in image, from the k number of centroids and the linear prediction order, the total computational cost is proportional to the product of width · height · k · r. Due to that, in one of the earliest solutions of this kind, a fixed dictionary with only 8 centroids was used, where each block of 8 × 8 px was assigned one of 8 fixed models that produced the smallest absolute error [32]. Header information associated with this block required 3 bits, thanks to which an index of prediction model (centroid) was identified. Later solutions introduced the minimization of mean square error as a method for calculating the best set of prediction coefficients, with grouping blocks into clusters with similar features, which used single shared prediction model [30]. Using vector quantization techniques (and fuzzy quantization [33]), an optimized sets of e.g. 16 prediction models were created. Thus, even with a high prediction order, the header section size did not increase significantly. A similar approach is used in this paper.

In [33], various block sizes were analyzed, and the best results were obtained with dimensions of 8 × 8 pixels. Therefore, both in our solution and in studies [30], [33], and [35], this size was also used (B_S = 8). In contrast, [35] considered different numbers of centroids (k = {16, 32, 64}), attempting to select a single compromise value. The authors of [28] approached this issue somewhat more precisely, proposing for the MRP codec a different number of classes depending on image size. Following the same guidelines, we also specified in Table 1 a set of parameters {k, r, n_b} for three image resolutions (based on a comprehensive series of experiments, using a training image set selected to cover a fairly wide range of image features).

METHODS OF DICTIONARY INITIALIZATION

Target function definition

Unlike lossy compression, our vector quantization solution used for lossless image compression uses significantly fewer centroids, while using centroids with more r elements (the k/n ratio is much smaller). This makes it more difficult to select k centroids that offer the most efficient data compression. Therefore, appropriate initialization of the dictionary is an important task, as it affects the final set of centroids representing a certain local minimum of the target function.

Equation 5 can be used as the target function, which uses the global entropy $H(S)$ of the

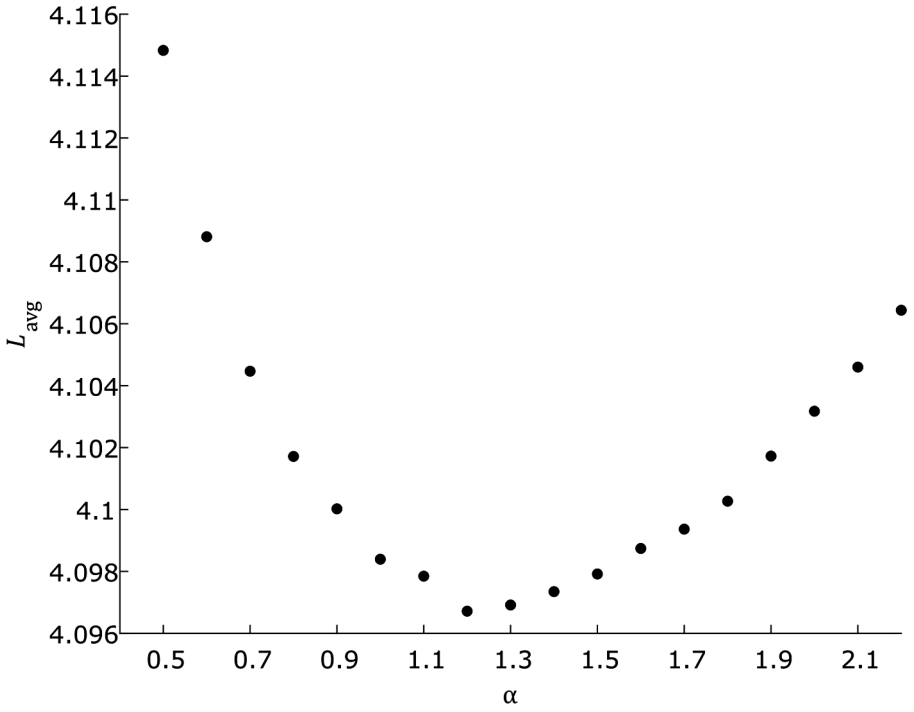

Figure 3. Results of bit averages for different values of α measured on images with resolution 512 × 512 px using proposed codec (VQ-MMAE$_2$)

prediction errors of the entire image. Although in the literature it is generally accepted that minimization of the mean square error tends to minimize the entropy of the set of prediction errors, it cannot be said that achieving the minimal mean square error guarantees a global minimum of $H(S)$. The results depend largely on the degree of fitting to the local data and the entire data processing flow. The experimental analysis of this subject can be found in [16]. In our solution, instead of MMSE, we proposed minimization of the minimum mean absolute error (MMAE) as a more advantageous approach for determining linear prediction coefficients. To achieve this, we used our own method to calculate the approximate value of the minimum mean absolute error. The aim is to better align the probability distribution of prediction errors with the geometric distribution required by the Golomb code, which is used for the initial compression of prediction errors. The output bits of Golomb coder are then encoded using context adaptive binary arithmetic coder (details in [34]).

To efficiently assign the i-th centroid to a specific block, a local target function is required in which the value $H(S)$ is not used. Therefore, at the step of choosing best centroid for given square 8 × 8 px (vector $\mathbf{x}_j$) we use Equation 6 which is a measure based on Minkowsky distance with an experimentally chosen power $\alpha = 1.2$ (instead of 2, as in classical solutions). Influence of α on bit average is presented in the Figure 3.

$$d_i = \frac{1}{z}\sum_{i=1}^{z}|e_j|^{\alpha} \tag{6}$$

where: $z = B_S^2$ denotes numer of pixels of each square.

Predictor coefficient calculation

In the proposed solution, each of the k classes (a set of squares with similar features) is assigned an individually tailored linear predictor of order r. In [35], it was demonstrated that minimizing the mean absolute error (MMAE) allowed for better results compared to MMSE, particularly when dividing the image into squares of size 8 × 8 px. Therefore, the proposed solution also uses the calculation of predictors based on MMAE. However, unlike in [35], in which an improved simplex algorithm was used, in our solution, we use an iterative reweighted least squares (IRLS) algorithm with lower computational complexity, which is based on classic WLS minimization:

$$\|e\|_1 = \sum_{m\in Q}\frac{1}{|e_m^{\text{old}}|}\cdot|e_m|^2 \approx \sum_{m\in Q}|e_m| \tag{7}$$

The algorithm is iterative and requires several MMSE minimizations. In the first iteration, the weights $w_m^{\,2} = e_m^{\,old}$ are set to 1, and in subsequent iterations, they represent the prediction error values obtained after encoding with the predictor from the given i-th iteration.

To calculate coefficients vector $\mathbf{B}$ for h-th class, in each iteration it is necessary to solve matrix equation (with Cholesky decomposition):

$$B = R^{-1}\cdot P \tag{8}$$

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

$$\mathrm{R}(j,i) = \sum_{m\in Q} w_m^2 \cdot y_m(i)\cdot y_m(j) \tag{9}$$

and $i = \{1, 2, ..., r\}$, $j = \{1, 2, ..., r\}$, and $\mathbf{P}$ is vector $r \times 1$ of elements

$$\mathrm{P}(j) = \sum_{m\in Q} w_m^2 \cdot x_m \cdot y_m(j) \tag{10}$$

where: Q denotes a set of all pixels of all squares belonging to a given h-th class, x_m determines the value of the next m-th encoded pixel, and $y_m(j)$ is the element of the vector $\mathbf{Y}_m = [y_m(1), y_m(2), ..., y_m(\mathrm{r})]$.

In proposed solution, a prediction model is using r closest pixels to the x_m (see Figure 1); therefore, the vector $\mathbf{Y}_m = [P_m(1), P_m(2), ..., P_m(r)]$.

Vector quantization algorithm

The step preceding quantization is the initialization of a dictionary. It requires assigning an individual predictor to each image block (an 8 × 8 px square) using MMAE method (10 iterations of IRLS). These predictors form the basis for creating the initial set of k centroids, which adapt to the characteristics of the encoded blocks during the reclassification stage. The initialization is described in section Simple approaches to vector quantization.

The quantization algorithm (Algorithm 1) requires several cycles of image encoding with each of k centroids (predictor of a class). Blocks

Repeat steps 1–2 for t times:
1. Reclassification of blocks to the closest class according to the criterion (6).
2. Calculating centroids based on the current assignment of blocks to classes using MMAE (3 IRLS iterations).

Algorithm 1. Quantization algorithm

are assigned to the most matching class based on formula (6) – the lower the value, the better the fitting. After reclassification of all of the blocks, centroids are recalculated (using Equation 8), and the process is repeated t times.

Simple approaches to vector quantization

In the classic k-means method, it is necessary to define a set of k centroids that serve as the initial form of the dictionary. There are many different approaches, although the most common is to randomly select k vectors $\mathbf{x}_j$ from an n-element set. Usually, it is assumed that after running k-means algorithm for a relatively large number of iterations, results are quite similar regardless of a randomly chosen set of initial centroids. Although there are special approaches to efficient dictionary initialization, for example, the k-means++ algorithm, which requires the use of k preliminary iterations to construct the initial form of the dictionary consisting of k centroids. For instance, for $k = 32$, the initialization time may be comparable to the time required by the actual quantization procedure, while in k-means codec designed by us we use only 20 iterations. Additionally, if we consider that due to the randomness present in the k-means++ algorithm, it is recommended to run such initialization multiple times, which further significantly increases its overall computational complexity. For this reason, there is a need to design a dictionary initialization method with the lowest possible computational complexity. However, considering the high computational complexity, it is suggested to apply heuristic approaches for dictionary initialization because the required number of k-means iterations should decrease significantly.

Let us consider the simplest case of lossless image compression with data segmentation, for which we will propose several methods for initial division of the set of pixels of encoded image into $k = 2$ classes.

Method I

We can create first 2 centroids based on the characteristics of individual pixels (instead of whole squares 8 repeat steps 1–2 for t times – reclassification of blocks to the closest class according to the criterion (6) – calculating centroids based on the current assignment of blocks to classes using MMAE (3 IRLS iterations)·8 px (instead of whole squares 8×8 px). To achieve this, a global linear predictor $\mathbf{B}$ of order $r = 36$ is calculated (Equations 8–10, where area Q consists of all pixels of image). Next, using vector $\mathbf{B}$ we determine the prediction errors and divide the pixel membership into two classes according to the sign of the value: negative and non-negative prediction errors. For each class an individual predictor is calculated (also using IRLS algorithm) giving two individual prediction coefficient vectors $\mathbf{B}_1$ and $\mathbf{B}_2$, which become the initial centroids for the k-means method.

Method II

Surprisingly good results can be achieved by dividing image into two equally wide columns. Then, similarly to *Method I,* an individual prediction coefficient vectors $\mathbf{B}_1$ and $\mathbf{B}_2$ are calculated, and are used as a starting centroids for k-means algorithm. This initialization considers the individual features of large areas of the image. Much worse results will be achieved, if the image will be divided into 8 × 8 px squares, and then divided into two sets following a chessboard pattern (the white and black squares on the chessboard). It is a simulation of pseudo-random assignment of individual squares to classes, which confirms that random selection as initialization in the k-means algorithm is not a good solution.

Method III

Another approach to dictionary initialization is to treat data from the beginning as a set of 8 × 8 px squares. Using Equations 11, 12, 13 and a rule used for division (defined by Equation 14), the squares are divided into two groups and predictors of two classes are calculated.

$$\bar{\Delta} = \frac{1}{n} \cdot \sum_{i=1}^{n} \Delta_i \tag{11}$$

$$\Delta_i = \sum_{j=1}^{r} \bar{d}_j \cdot \left| b_j^{(i)} + \bar{b}_j \right|^2 \tag{12}$$

$$\bar{d}_j = \frac{1}{\sqrt{\left(\Delta x_j\right)^2 + \left(\Delta y_j\right)^2}} \tag{13}$$

where: n – number of squares in image, i – square index, $\bar{\mathbf{b}}$ – average predictor calculated from all n individual predictors, $\mathbf{b}^{(i)}$ – individual predictor of i-th square.

Equation 13 is the inverse of the Euclidean distance between pixel x_m and its j-th neighbors shown in Figure 1.

$$\begin{cases} class \leftarrow c_1, \; if \; \Delta_i > \beta \cdot \bar{\Delta} \\ class \leftarrow c_2 \; \; if \; \Delta_i \leq \beta \cdot \bar{\Delta} \end{cases} \tag{14}$$

Value of β was chosen experimentally from tested range {0.5, ..., 1.3} and set to 0.7 after testing behavior of a binary division using *Method III* only (using proposed codec, which after initialization stage performs $t = 20$ iterations of vector quantization). Results of these experiments are shown in Figure 4.

Method IV

In this proposition, the binary quantization method for dividing a set of squares into 2 centroids is based on the threshold value $\bar{b}_i$, which is an average value of i-th prediction coefficient (from all n individual predictors). The k-means quantization is performed in two iterations (this is a case of binary scalar quantization), where the threshold initialization is based on the mean of the given i-th coefficient, and then the following steps are performed iteratively:

a) centroid calculation in each class,
b) reclassification of squares according to the Euclidean distance (where it is closer to each individual i-th coefficient).

During this process centroids are calculated as the arithmetic mean of individual predictors of squares assigned to given class. This algorithm continuously works on two sets in which reclassification and centroid recalculation are performed.

Method V

The coefficients b_1 and b_2 of all individual predictors are plotted on a plane as points with coordinates (b_1, b_2), forming a cloud that is divided into two groups by:

- linear regression line,

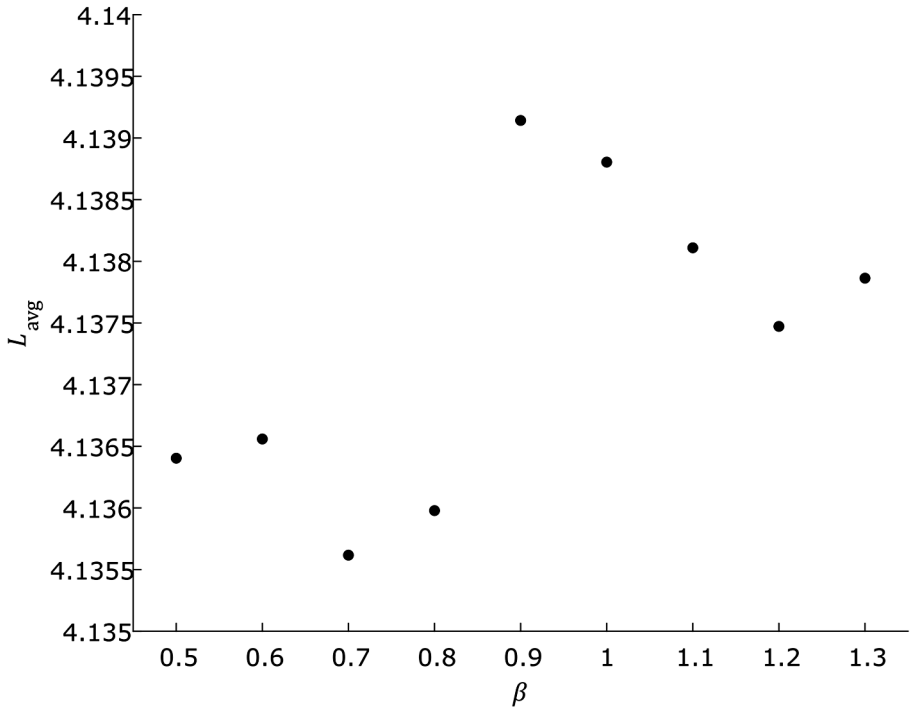

Figure 4. Results of bit averages for different values of β obtained during tests of *Method III* as a binary division method

- line perpendicular to the linear regression line that passes through the point $C_0(\bar{b}_1, \bar{b}_2)$.

The linear regression $f(x) = ax + b$ is calculated using Least Squares method, where the coefficients a and b are calculated with:

$$a = \frac{m\sum_{i=1}^{m} x_i y_i - \sum_{i=1}^{m} x_i \sum_{i=1}^{m} y_i}{m\sum_{i=1}^{m} x_i{}^2 - \left(\sum_{i=1}^{m} x_i\right)^2} \quad (15)$$

$$b = \frac{\sum_{i=1}^{m} x_i{}^2 \sum_{i=1}^{m} y_i - \sum_{i=1}^{m} x_i y_i \sum_{i=1}^{m} x_i}{m\sum_{i=1}^{m} x_i{}^2 - \left(\sum_{i=1}^{m} x_i\right)^2} \quad (16)$$

where: m denotes number of points, x_i and y_i denote values of coefficients of given point.

An example of a cloud created for the Lennagrey image is shown in Figure 5. The solid line represents the linear regression line and the dashed line is perpendicular to it. Depending on the algorithm variant, the cloud is divided into two classes according to one of two lines, and the axes in the coordinate system can be switched from $X = b_1$, $Y = b_2$ to $X = b_2$, $Y = b_1$.

Method VI

Initially, all squares forming the image belong to a single class encoded by predictor $\mathbf{C}_1$, which is a global predictor (for the entire image). The image is encoded using predictor $\mathbf{C}_1$ and individual predictors of the squares. Then, for each square, the following value is calculated:

$$q = \frac{\frac{1}{m}\sum_{i=1}^{m} |e_i^h|}{\frac{1}{m}\sum_{i=1}^{m} |e_i^{IND}| + 1} \quad (17)$$

where: the numerator contains prediction errors obtained by encoding with predictor of class to which the block belongs, and the denominator contains the prediction errors obtained by encoding with the individual predictor. The values of q are then sorted and divided into two classes based on the median value.

Table 2 shows the results for each method collected for a set of 26 test images with a resolution of 512 × 512 px from the image database [36]. These are the results of the initialization itself (because the full compression algorithm after the centroid initialization stage is completed performs a clearing of block assignments to classes, one iteration of reclassification is run based on Equation 6 to reassign blocks to classes).

Figure 6 shows the convergence of bit average to the best result measured over 20 quantization iterations. The vector quantization algorithm using initialization *Method I* converges to the best result

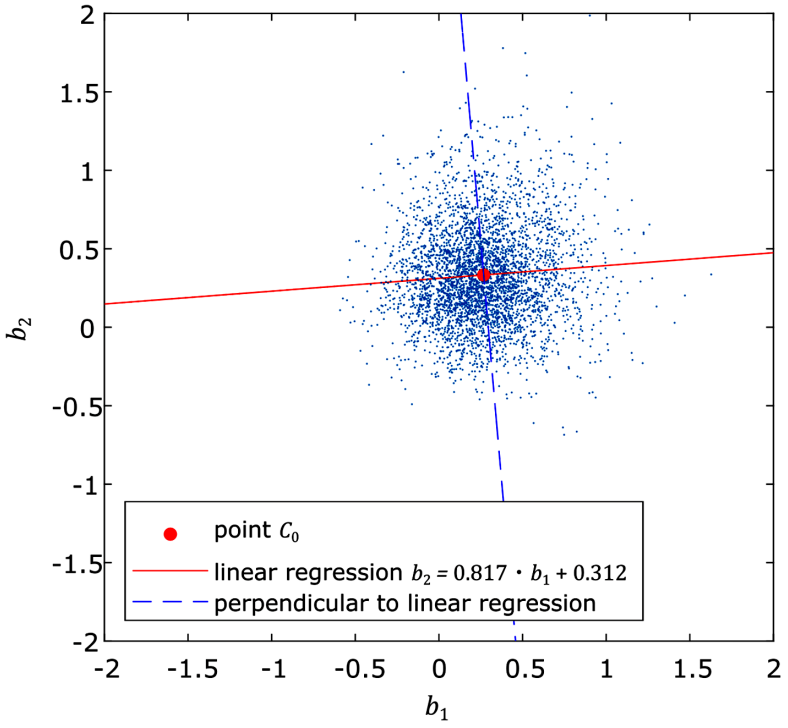

Figure 5. Cloud of points and its division using *Method V* based on coefficients b_1 and b_2 on the example of a Lennagrey image

Table 2. Results of proposed initialization methods for set of 26 images with resolution 512 × 512 px

Initialization method	Entropy [bits/pixel]	Bit average [bits/pixel]
I	4.4103098	4.1766487
II – columns	4.3819838	4.1563932
II – chessboard	4.4065037	4.1768435
III	4.3835328	4.1574766
IV i = 1	4.3677179	4.1460818
IV i = 2	4.3658215	4.1444784
V a)	4.3672637	4.1457003
V b)	4.3674585	4.1457743
VI	4.3711289	4.1498566

more slowly, and although both methods yield quite similar results in later iterations, choosing *Method IV* as the initialization method produces better results with fewer iterations. Therefore, it is the more advantageous method because of the possibility of reducing the number of iterations, which leads to time optimization of designed solution. The summary of advantages and disadvantages of described initialization methods is provided in Table 3.

The final initialization algorithm proposed in this work combines all the discussed binary partitioning methods into a single solution, in which

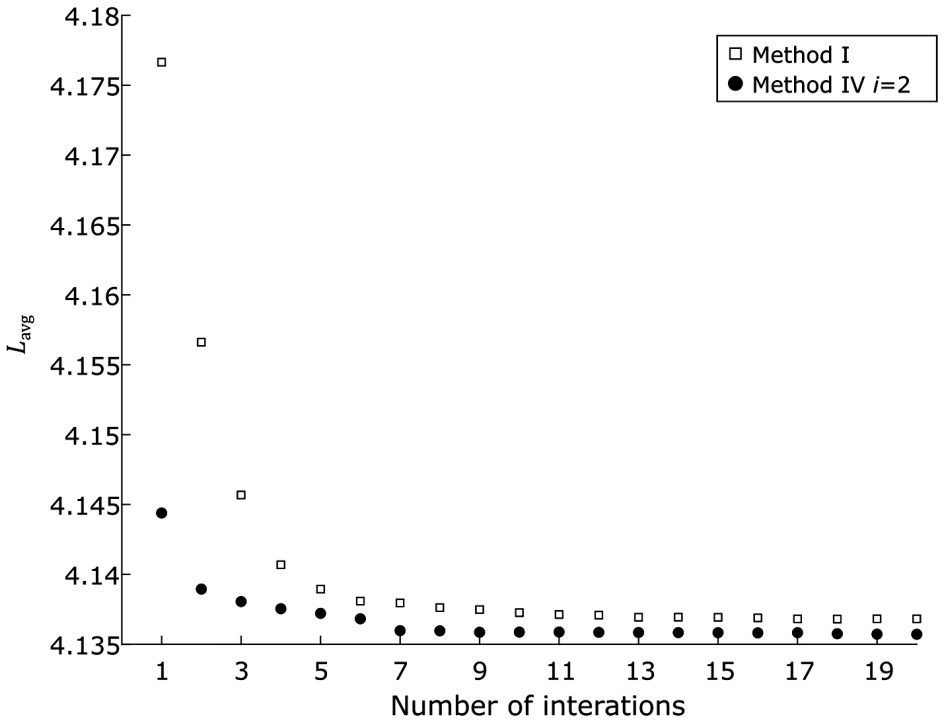

Figure 6. Convergence of bit average L_{avg} to the minimum for vector quantization using initialization Method I and Method IV measured over 20 iterations

Table 3. Advantages and disadvantages of initialization Method I-VI

Method	Advantages and disadvantages
I	– Simple criteria for class division.
II	– In case of „chessboard" pattern, the randomness of assignment causes a need for many reclassifications of squares. – Division into equally wide columns considers individual features of large areas and leads to quite good initialization.
III	– Computationally complex.
IV	– Tunes the threshold value to achieve better classification. – Simple criteria for class division. – Requires 2 iterations (with centroids recalculation).
V	– Computationally complex. – Must be performed as a first division in the binary tree based initialization algorithm. – Offers good classification.
VI	– Computationally complex.

First 4 classes (*Method V*):
1. Set as axis as axis X $= b_2$, axis Y $= b_1$.
2. Calculate point $C_0(\bar{b}_1, \bar{b}_2)$.
3. Calculate regression line and line perpendicular to it going through point C_0 (Equation 15, 16).
4. Use each quarter to create first set of classes (4 classes).
Binary division of 4 classes from previous step resulting in 8 leaves (*Method IV*)
5. Calculate $\bar{b}_3$ for each class.
6. Divide a set of squares into another 2 classes based on threshold $\bar{b}_3$.
For 2 times perform binary scalar quantization with threshold $\bar{b}_3$ tuning:
7. Perform squares reclassification according to the Euclidean distance (where it is closer to each individual 3-th coefficient).
8. Recalculate centroids.
Binary division of 8 classes from previous step resulting in 16 leaves (*Method VI*).
9. Divide classes using *Method VI* (Equation 17) producing 16 classes in total.
Binary division of 16 classes from previous step resulting in 32 leaves (*Method III)*.
10. Calculate predictor C_1, which is a global predictor (for the entire image).
11. Divide classes using with Equation 11 to 14 with $\beta = 0.7$.

Algorithm 2. Binary tree based initialization algorithm for scenario of k = 32 classes

Table 4. Comparison of bit averages of classical initialization approach and proposed initialization algorithm (VQ-MMAE$_2$) for a configuration "with" and "without" reclassifications

Parameter	Bit average (without reclassification t = 0)				Bit average (with t = 0 reclassificaiton)			
Image	Minimum	Maximum	Average	VQ-MMAE$_2$	Minimum	Maximum	Average	VQ-MMAE$_2$
Aerial	4.65170	4.68011	4.66637	4.61499	4.55579	4.56445	4.56060	4.56021
Airfield	4.88953	4.90207	4.89733	4.87100	4.84860	4.85080	4.84952	4.84833
Airplane	3.68304	3.70029	3.68926	3.65302	3.63086	3.63361	3.63229	3.63147
BaboonTMW	5.79288	5.82236	5.80652	5.75278	5.72806	5.73187	5.73031	5.72916
Barb512	4.33713	4.38004	4.36469	4.22400	4.15671	4.17816	4.16526	4.15112
Boat512	4.52979	4.56232	4.54466	4.48834	4.44656	4.45218	4.44974	4.45013
Bridge	3.42209	3.43643	3.42980	3.40500	3.37857	3.38275	3.37974	3.38022
Couple	4.17145	4.19806	4.18331	4.13217	4.10156	4.10703	4.10431	4.10556
Crowd512	3.66730	3.69656	3.67852	3.62564	3.60205	3.60748	3.60434	3.60342
Elaine	4.42575	4.44479	4.43736	4.39752	4.37061	4.37366	4.37234	4.37119
Finger	5.24802	5.25613	5.25246	5.25342	5.22473	5.22720	5.22549	5.22565
Frog512	5.03766	5.04794	5.04366	5.02744	5.00900	5.01236	5.01056	5.00879
Goldhill	4.56216	4.59122	4.57466	4.52686	4.50027	4.50348	4.50164	4.49930
Harbour512	4.34531	4.39197	4.36867	4.31833	4.27930	4.28918	4.28416	4.27548
Lax512	5.61298	5.62415	5.62015	5.58188	5.55750	5.56046	5.55903	5.55923
LenaTMW	4.43561	4.46030	4.44756	4.38342	4.34726	4.35489	4.35009	4.34879
Lennagrey	4.04083	4.06091	4.05136	3.98798	3.94980	3.96390	3.95501	3.95303
Man512	4.34393	4.35837	4.35230	4.28308	4.26276	4.26501	4.26367	4.26276
PeppersTMW	4.32532	4.35703	4.34188	4.29498	4.26532	4.27048	4.26780	4.26889
Sailboat	4.52991	4.55161	4.54013	4.50204	4.48044	4.48947	4.48381	4.48389
Seismic	2.11014	2.12549	2.12029	2.11887	2.08820	2.09250	2.09113	2.08969
Shapes	1.07706	1.17700	1.13068	1.07532	0.99957	1.02200	1.01274	1.00992
Tank512	3.83322	3.84613	3.84069	3.82788	3.81082	3.81226	3.81170	3.81177
Truck512	4.08868	4.10782	4.09800	4.08859	4.07159	4.07391	4.07287	4.07251
Woman1	3.90009	3.93527	3.91905	3.85638	3.81534	3.82184	3.81794	3.81693
Woman2	3.05200	3.06842	3.05962	3.02716	2.99841	3.00314	3.00117	2.99719
Average	4.15821	4.18395	4.17150	4.12762	4.09537	4.10170	4.09836	4.09672

the division occurs gradually in accordance with the principle of constructing a complete binary tree, where each higher-level class is split into two using next method until the specified number of k classes is created. The first stage of division is done using *Method V*, with the difference that a set of 4 classes is created at once instead of two. The axes of the coordinate system are set as axis $X = b_2$, axis $Y = b_1$, and the division itself is made simultaneously by the regression line and the line perpendicular to it, so each quarter visible in Figure 5 forms a separate class. Further binary division on these 4 classes into binary tree leaves is done using *Method IV* with coefficient b_3 (b_1 and b_2 are already used in *Method V*) – resulting in 8 classes in total. Depending on the required total number of k classes (see Table 1), in the next stages of building binary tree, the next methods are *Method VI* and *III* consecutively, which leads to trees with 16 and 32 leaves respectively. Algorithm 2 presents scenario for building $k = 32$ classes. If a lower number of classes is needed, the further division is stopped. After completing this hierarchical initialization, 20 iterations of the classic k-means algorithm adapted for lossless image compression are performed.

A classical approach to dictionary initialization is to randomly select k vectors from the entire set of data vectors (e.g., blocks in the case of lossy compression). However, in lossless compression, where the elements of the dictionary are not centroids representing groups of vectors with similar characteristics but prediction coefficients, k blocks are randomly selected. Then, for each of these blocks, an r-th order linear predictor is calculated using the MMAE method, whose coefficients become the initial values of the i-th centroid. It is assumed that the vector quantization (k-means) algorithm converges relatively well to a local minimum if a sufficiently high number of iterations is used. Table 4 compares the bit averages achieved by this classical approach (because the algorithm is stochastic, each image was encoded 10 times) with the results of the dictionary initialization algorithm proposed in this paper. Columns 2, 3, and 4 present the minimum, maximum, and average bit averages (from 10 tests) for a given test image. Column 5 contains the results obtained after applying the initialization according to Algorithm 2 (VQ-MMAE$_2$ codec). In columns 2–5, the number of vector quantization iterations is set to t = 0 to demonstrate the advantage of the proposed method over the classical initialization approach. In columns 6 through 9 the same experiment is repeated, but the number of k-means iterations is set to $t = 20$. Although in this case the advantage of using Algorithm 2 over classical initialization is smaller, it is still noticeable. However, the key aspect is the significant acceleration of k-means convergence owing to heuristic initialization.

The algorithm used in our solution, just like in MRP 0.5, is k-means, which is a specific case of fuzzy quantization (fuzzy c-means), offering slightly lower computational complexity compared to the generalized form. Although multiple iterations are possible with fuzzy c-means, in the final stage it is necessary to additionally perform hard classification, based on the dominant membership value of a square to a given centroid. Further research is planned to examine this approach.

METHODS FOR IMPROVING COMPRESSION EFFICIENCY

Header data compression

Compression of the shades occurring in the image

In the header section of output file, the information about shades present in the original file is embedded. For 8-bit grayscale images it requires saving the information about occurrence of all 256 possible shades. For this, it is necessary to construct a 256-bit occurrence map, where setting the corresponding bit indicates the presence of a given shade in the image. This sequence can be also compressed, by passing it into binary adaptive arithmetic coder. In the header one byte is reserved to save information about total number N_{grey} of shades of original image. Using this

Compression of the centroid numbers assigned to the consecutive squares

In order to decrease the size of header section, where the information about the assignment of centroid to each square (number from 0 to $k-1$) is stored, a compression of these centroid numbers is introduced using an adaptive arithmetic coder. Centroids are sorted in descending order according to the number of occurrences (number

of squares using a centroid), and new indices are assigned to them. Arithmetic coder is initialized with a geometric distribution with $f = 4$:

$$N_c(i) = \lfloor f \cdot 0{,}75^i \rfloor + 1 \tag{18}$$

where: i – number between 0 and $k - 1$, f – amplitude initializing vector of occurrences.

After encoding consecutive indices, the distribution is updated, to adjust to the local characteristics of encoded image. After total count of occurrences exceeds threshold of 2^8, all elements from vector of occurrences are halved (which is a cyclic forgetting mechanism).

Prediction errors mapping

Although the range of prediction errors is <–255; 255>, in special cases it is possible to reduce it when an assumption is made that predicted value cannot be negative or greater than $N_{grey} - 1$. The proposed solution uses an algorithm to reduce the range of prediction errors described in detail in [34]. Reduced range can have a positive impact on improving the efficiency of adaptive coding. Probability distribution symmetry of predicted values $\hat{x}_m$ varies depending on x_m. For example, a distribution for $x_m = 40$ is shown in the Figure 7, where the part forming long tail and goes beyond the symmetry is marked in red (for values greater than 80). To transform the distribution into symmetric and decrease mean absolute error, the values from „tail" can be casted alternatingly (odd and even values are placed on the extensions of both blue lines).

Constant component removal

Although linear prediction tailored to the local area (8 × 8 px square) offers high efficiency in predicting the value of $\hat{x}_m$, it is possible to make an additional adjustment. In many solutions (e.g. in JPEG-LS and CALIC codecs) it is proposed to utilize adaptive method for constant component removal (bias cancelation), also known as context-based error correction techniques. By introducing additional block Context-Dependent Constant Component Removing (CDCCR) which removes constant component C_{mix} associated with a certain context, it is possible to further improve prediction efficiency. Removing a C_{mix} associated with one of the 1024 contexts results in a slight modification of the Equation 2:

$$e_m = x_m - [\hat{x}_m + C_{\text{mix}}] \tag{19}$$

where: $\hat{x}_m$ is determined as a predicted value based on a linear prediction tuned to a given square (class).

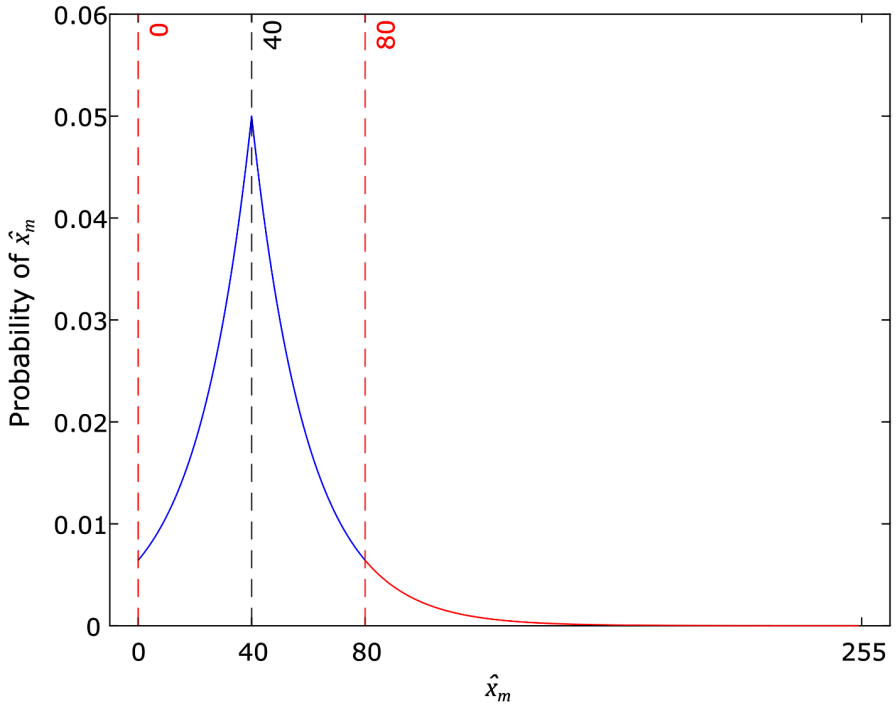

Figure 7. Theoretical probability distribution of predicted values $\hat{x}_m$ for $x_m = 40$. The "long tail" located outside region of symmetry is marked in red

Other details, including the remaining stages of prediction error encoding (Golomb code and context adaptive arithmetic coder) and the data processing flow used for encoding single pixel presented in the Figure 8, which are used in proposed solution are described in [34].

Compression efficiency

The most well-known example of a codec with a short decoding time, which is based on vector quantization is MRP 0.5 [28]. For MRP 0.5, the encoding time of Lennagrey image with a resolution of 512 × 512 px is 565.584 s. (in optimized mode "-o"). Result achieved using AMD Ryzen 7 5700x 3.4GHz, 32GB RAM, Windows 10, compiler MinGW 11.0 (-funroll-loops -O4), single threaded code. Although there are extensions of this codec, namely xMRP [37], GPR-BP [38], and MRP-SSP [39], they are characterized by even higher implementation complexity of the encoder (even longer encoding time). Our solution named VQ-MMAE$_2$ [40], simplifies the entire quantization process compared with the method presented in [41] (in Tables 6 and 8, it is presented under the name VQ- MMAE$_1$). This is due to the introduction of efficient dictionary initialization needed by vector quantization. Encoding time of an image with 512 × 512 px resolution has decreased from 263.837 s. to 178.213 s, which is an improvement of 32%. It should be emphasized that owing to the proposed method in VQ-MMAE$_2$, the total number of iterations of *k*-means algorithm is reduced from 48 to 20 compared with VQ-MMAE$_1$. Currently, the dictionary initialization time accounts for 10.91% of the total image compression time, whereas the main dictionary construction procedure (*k*-means) requires 74.74% of the time. The remaining 14.2% is taken up by loading and preliminary data preparation, and only 0.15% by the final output file generation procedure (final compression). Assuming that N is the number of image pixels, a single iteration of *k*-means requires $N \cdot r \cdot k$ multiplications and additions to be performed. Therefore, the complexity of one *k*-means iteration is $O(N \cdot r \cdot k)$, whereas the complexity of hierarchical initialization, considering all its stages, can be estimated as less than 3 *k*-means iterations. Despite the significant acceleration of encoding, the compression efficiency has further increased, as shown in Tables 5–10. Therefore, the proposed solution is more than three times faster than MRP 0.5 optimized (command line argument "-o" offering the highest efficiency). In addition, the advantage of VQ-MMAE$_2$ is a relatively short decoding time of 0.335 s. The proposed solution, similar to MRP 0.5 presented 20 years earlier, is characterized by a long encoding time. The optimization of the code (including its parallelization) is planned for the next phase of codec development. Combined with the ever-increasing performance of computers, this may eventually result in the final version of VQ-MMAE offering an acceptable execution time for archiving image databases, where multiprocessor systems are capable of encoding many images simultaneously in parallel. There are also plans to implement reversible image

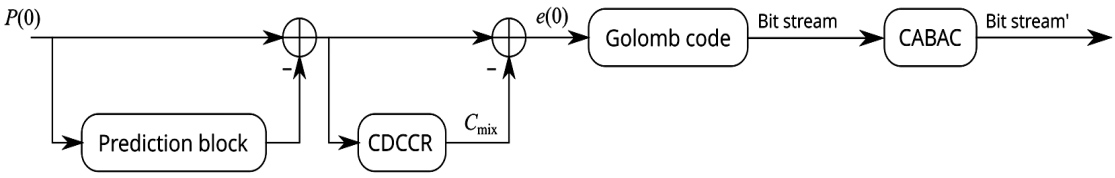

Figure 8. Compression scheme used in proposed codec

Table 5. Bit average based on a database of 45 standard test images for different codecs [36]

Codec	WebP [43]	WebP2 [44]	Pngcrush [45]	JPEG-XL[46]	AVIF [47]	FLIF [48]
Average	4.205	4.269	4.719	4.123	4.802	4.324

Table 6. Table 5 (continuation with results of other codecs)

Codec	MRP [49]	MRP optimized	VQ-MMAE$_1$	VQ-MMAE$_2$
Average	4.058	4.022	3.983	3.971

transformation for the purpose of color image compression, thanks to which it is possible to independently and simultaneously encode the luminance and chrominance components in a highly efficient manner. An analysis of this type of transformation is discussed in [42]. Table 9 and 10 presents a comparison of bit averages of not only classical codecs (the non-learning category), but also those representing the deep learning category (L3C, CWPLIC, LCIC duplex). Based on this list, it can be observed that the solution we propose offers an average of 7.22% lower bit average compared to JPEG-LS. Both the training dataset and the test image sets

Table 7. Bit average results for 16 images with various characteristics (from set of 45 images)

	Codec					
Image	WebP	WebP2	Pngcrush	JPEG-XL	AVIF	FLIF
Airfield	5.09	5.08	5.82	4.92	5.10	6.04
Balloon	2.90	2.92	3.25	2.76	2.87	3.05
Bridge	3.58	3.54	4.26	3.42	3.62	6.09
Couple256	3.71	3.68	4.15	3.56	3.78	4.03
Earth	2.94	2.92	3.37	2.83	2.99	3.58
Elif	3.17	3.14	3.60	3.00	3.18	3.51
Frog512	3.53	6.90	7.05	6.85	6.94	7.95
Gold	4.46	4.45	4.68	4.35	4.51	4.73
Lennagrey	4.13	4.13	4.60	4.03	4.28	4.40
Noisesquare	5.09	5.09	5.76	5.24	5.34	5.62
Seismic	2.95	2.92	3.14	2.73	2.79	3.25
Sensin	3.75	3.74	4.25	3.53	3.72	4.16
Shapes	1.00	0.99	1.18	0.71	0.83	1.21
Tank512	3.97	3.96	4.88	3.82	3.95	5.04
Truck512	4.18	4.14	4.76	4.00	4.21	4.90
Woman2	3.20	3.22	3.68	3.05	3.22	3.46

Table 8. Table 7 (continuation with results of other codecs)

	Codec			
Image	MRP	MRP optimized	VQ-MMAE$_1$	VQ-MMAE$_2$
Airfield	5.35	5.32	4.86	4.85
Balloon	2.60	2.58	2.66	2.66
Bridge	5.34	5.28	3.38	3.38
Couple256	3.41	3.39	3.51	3.50
Earth	3.06	3.04	2.86	2.85
Elif	2.63	2.62	2.67	2.66
Frog512	5.94	5.90	5.02	5.01
Gold	4.23	4.21	4.28	4.27
Lennagrey	3.91	3.89	3.96	3.95
Noisesquare	5.31	5.27	5.31	5.26
Seismic	2.07	2.05	2.10	2.09
Sensin	3.10	3.09	3.14	3.14
Shapes	0.76	0.70	1.03	1.01
Tank512	4.66	4.64	3.82	3.81
Truck512	4.42	4.40	4.08	4.07
Woman2	2.94	2.92	3.00	3.00

included images with various characteristics (natural ones from cameras and digital devices, with low and high levels of noise, as well as artificially generated ones). The proposed solution is mainly targeted at images obtained from digital cameras (with varying levels of noise), similarly to the commonly used WebP codec, as confirmed by the results for individual test images compared to JPEG-LS. For example, for Barbara image (with textured fragments), our codec achieved a bit average lower by 16.38%. However, for artificially generated images such as Comic (with many smooth tonal transitions and no noise), the advantage of VQ-MMAE$_2$ is no longer as noticeable, and in the case of both WebP versions, the result was actually worse compared to JPEG-LS. The tests did not reveal any case in which VQ-MMAE$_2$ was inferior to JPEG-LS (Table 9 and 10).

From the set of 45 test images used in Tables 5 and 6, 16 images with various characteristics and resolutions were selected, with detailed results presented in Tables 7 and 8. This allows for a more in-depth demonstration that some codecs perform better or worse, depending on the features of the image. For example, not every codec uses the fact that some colors from the full 256-value palette do not appear in

Table 9. Bit average based on a test image dataset from paper [22]

	Codec							
Image	BPG	PNG	LCIC	JPEG 2000	JPEG-LS	JPEG-XL	AVIF	FLIF
Airplane	4.32	4.26	3.99	4.00	3.80	3.71	4.18	3.82
Barbara	5.06	5.22	4.61	4.61	4.70	4.40	4.95	4.55
Coastguard	5.70	5.06	4.82	4.83	4.86	4.73	5.21	4.92
Comic	6.15	5.84	5.63	5.65	5.30	5.07	6.21	5.50
Flowers	5.18	5.08	4.91	4.92	4.62	4.51	5.18	4.73
Goldhill	4.95	4.70	4.58	4.59	4.43	4.37	4.75	4.50
Lennagrey	4.54	4.61	4.31	4.31	4.24	4.16	4.40	4.28
Mandrill	6.61	6.23	6.11	6.11	6.04	5.98	6.62	6.14
Monarch	4.10	4.26	3.82	3.82	3.70	3.54	4.02	3.66
Pepper	4.77	4.90	4.63	4.63	4.51	4.48	4.71	4.58
Ppt3	2.20	2.35	2.41	2.41	2.04	1.84	2.26	1.88
Zebra	5.83	5.19	4.89	4.89	4.81	4.66	5.38	4.85
Average	4.951	4.808	4.559	4.564	4.421	4.288	4.823	4.448

Table 10. Table 9 (continuation with results of other codecs)

	Codec						
Image	WebP	WebP2	L3C	CWPLIC	LCIC duplex	VQ-MMAE$_1$	VQ-MMAE$_2$
Airplane	3.86	3.84	4.56	3.69	3.69	3.63	3.61
Barbara	4.52	4.51	5.44	4.35	4.36	3.94	3.93
Coastguard	4.81	4.81	5.82	4.80	4.83	4.41	4.41
Comic	5.40	5.39	6.60	4.83	4.83	5.01	5.00
Flowers	4.72	4.71	5.53	4.41	4.35	4.38	4.39
Goldhill	4.46	4.42	5.27	4.33	4.33	4.22	4.22
Lennagrey	4.13	4.13	4.95	4.13	4.08	3.96	3.95
Mandrill	5.89	5.90	6.97	5.95	5.89	5.74	5.73
Monarch	3.70	3.72	4.37	3.40	3.45	3.42	3.41
Pepper	4.46	4.47	5.38	4.67	4.38	4.28	4.26
Ppt3	2.02	2.01	3.71	2.14	2.07	1.93	1.90
Zebra	4.85	4.85	6.08	4.65	4.68	4.36	4.41
Average	4.402	4.397	5.390	4.279	4.245	4.113	4.102

individual images (e.g., Airfield, Tank512). In such cases, even codecs with lower implementation complexity such as WebP or JPEG-XL perform better than MRP 0.5. A similar situation occurs with the Noisesquare image, which is an artificially generated image with a high level of added noise. However, in the vast majority of cases, the MRP and VQ-MMAE codecs are able to gain an advantage, as confirmed by the aggregate results from Tables 5 and 6.

In the case of specific categories, such as medical images, there are too many similarities between the individual images. Therefore, to avoid the well-known issue of overfitting known from neural networks, we ensured not to tune the codec parameters for this purpose. Nevertheless, encoding this type of images with VQ-MMAE$_2$ is not a problem.

CONCLUSIONS

Various lossless image compression methods have been developed over the past 30 years. They can be divided depending on used basic coding mechanisms or depending on the implementation complexity of both coder and decoder. In most cases, certain limits are set, especially those related to the execution time of the encoding and decoding.

In this paper, we proposed a solution based on dividing the image into blocks of 8 × 8 pixels and vector quantization. To increase compression efficiency a linear prediction was used with a fast algorithm to minimize absolute error called IRLS, and this approach is our original idea. Proposed codec is characterized by a relatively fast decoding time, while offering high compression efficiency. The achieved bit average is 7.22% lower than that of the widely known JPEG-LS. However, this comes at a considerable computational cost during encoding stage, which affects the compression time. This is due to the use of algorithms based on vector quantization. To prevent (to some extent) the negative effects from the user's perspective, a novel method of dictionary initialization was proposed. As a result, compared to the previous version of VQ-MMAE codec, a significant acceleration of computation (on average faster by 32%) was achieved, while simultaneously increasing compression efficiency, as confirmed by tests performed using a set of test images.

REFERENCES

1. Hussain A.J., Al-Fayadh A., Radi N. Image compression techniques: A survey in lossless and lossy algorithms. Neurocomputing. 2018; 300: 44–69. https://doi.org/10.1016/j.neucom.2018.02.094
2. Liang Y., Jia T., Li N., Liu X., Jiang J., Lu G., et al. Review of Static Image Compression Algorithms. In: 2024 IEEE 7th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC). Chongqing, China; 2024; 222–231. https://doi.org/10.1109/IAEAC59436.2024.10503622
3. D'Amato J.P., Oliveto M. Improving backup strategies in large DICOM databases based on weighted image compression. SN COMPUT SCI. 2025; 6(4): 337. https://doi.org/10.1007/s42979-025-03871-z
4. Yang Y., Sisk C.D., Calhoun J.C. Evaluating Lossy and Lossless Compression for DICOM Medical Files. In: 2024 IEEE International Conference on Big Data (BigData). Washington, DC, USA: IEEE; 15–18 December 2024; 4517–4523. https://doi.org/10.1109/BigData62323.2024.10825078
5. Gunasheela K.S., Prasantha H.S. Satellite Image Compression-Detailed Survey of the Algorithms. In: Proceedings of International Conference on Cognition and Recognition. Singapore: Springer Singapore; Lecture Notes in Networks and Systems, 2018; 14: 187–98. https://doi.org/10.1007/978-981-10-5146-3_18
6. Altamimi A, Ben Youssef B. Lossless and near-lossless compression algorithms for remotely sensed hyperspectral images. Entropy. 2024; 26(4): 316. https://doi.org/10.3390/e26040316
7. Sayood K. Introduction to Data Compression. Fifth Edition. San Francisco: Morgan Kaufmann; 2018.
8. Andriani S., Calvagno G., Darigon M., Rinaldo R., Knee M., Walland P., et al. Comparison of lossy to lossless compression techniques for digital cinema. In: International Conference on Image Processing. Singapore; 24-27 October 2004; 513–516. https://doi.org/10.1109/ICIP.2004.1418803
9. Mentzer F., Gool L.V., Tschannen M. Learning Better Lossless Compression Using Lossy Compression. In: Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; 6637–6646. https://doi.org/10.1109/CVPR42600.2020.00667
10. Rhee H., Jang Y.I., Kim S., Cho N.I. LC-FDNet: Learned Lossless Image Compression with Frequency Decomposition Network. In: Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR). New Orleans, LA, USA, 18–24 June 2022; 6023–6023. https://doi.org/10.1109/CVPR52688.2022.00594
11. Weinberger M.J., Seroussi G., Sapiro G. The LOCO-I

lossless image compression algorithm: principles and standardization into JPEG-LS. IEEE Transactions on Image Processing. 2000; 9(8): 1309–1324. https://doi.org/10.1109/83.855427

12. Wu X., Memon N. CALIC - A context based adaptive lossless image codec. In: Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing. 1996; 4: 1890–1893. https://doi.org/10.1109/ICASSP.1996.544819
13. Codec WebP. An image format for the Web, https://developers.google.com/speed/webp (Accessed: 14.09.2025)
14. Wu X., Barthel E.U., Zhang W. Piecewise 2D autoregression for predictive image coding. In: Proceedings of International Conference on Image. Chicago, Illinois, USA; 1998; 901–904. https://doi.org/10.1109/ICIP.1998.727397
15. Ye H., Deng G., Devlin J.C. Adaptive linear prediction for lossless coding of greyscale images. In: Proceedings of International Conference on Image Processing. 2000; 1: 128–131. https://doi.org/10.1109/ICIP.2000.900911
16. Ulacha G., Stasiński R., Wernik C. Extended multi WLS method for lossless image coding. Entropy 2020; 22(9): 919. https://doi.org/10.3390/e22090919
17. Weinlich A., Amon P., Hutter A., Kaup A. Probability distribution estimation for autoregressive pixel-predictive image coding. IEEE Transactions on Image Processing, March 2016; 25(3): 1382–1395. https://doi.org/10.1109/TIP.2016.2522339
18. Kau L.J., Lin Y.P., Lin C.T. Lossless image coding using adaptive, switching algorithm with automatic fuzzy context modelling. IEEE Proceeding Vision, Image and Signal Processing, October 2006; 153(5): 684–694. https://doi.org/10.1049/ip-vis:20045256
19. Marusic S., Deng G., Adaptive prediction for lossless image compression, Signal Processing: Image Communications, 2002; 17(5): 363–372. https://doi.org/10.1016/S0923-5965(02)00006-1
20. Takizawa K., Takenouchi S., Aomori H., Otake T., Tanaka M., Matsuda I., Itoh S., Loss-less image coding by cellular neural networks with minimum coding rate learning. In: Proceedings of 20th European Conference on Circuit Theory and Design (ECCTD), Linköping, Sweden, 29–31 August 2011; 33–36. https://doi.org/10.1109/ECCTD.2011.6043337
21. Rahman A., Hamada M., Rahman A. A comparative analysis of the state-of-the-art lossless image compression techniques. SHS Web of Conferences; 2022; 139. https://doi.org/10.1051/shsconf/202213903001
22. Rhee H., Jang Y.I., Kim S. Cho N.I. Lossless Image Compression by Joint Prediction of Pixel and Context Using Duplex Neural Networks. IEEE Access, 2021; 9: 86632–86645. https://doi.org/10.1109/ACCESS.2021.3088936
23. Hoogeboom E., Peters J.W.T., Berg R., Welling M. Integer Discrete Flows and Lossless Compression. In: Proceedings of 33rd Conference on Neural Information Processing Systems, December; 2019. https://doi.org/10.48550/arXiv.1905.07376
24. Salimans T., Karpathy A., Chen X., Kingma D.P. PixelCNN++: Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications. 5th International Conference on Learning Representations, Toulon, France; April 2017. https://doi.org/10.48550/arXiv.1701.05517
25. Kojima H., Kita Y., Matsuda I., Itoh S., Kameda Y., Unno K., Kawamura K. Improved Probability Modeling for Lossless Image Coding Using Example Search and Adaptive Prediction, In: Proceedings of SPIE 12177, International Workshop on Advanced Imaging Technology, April 2022. http://doi.org/10.1117/12.2625972
26. Zhang S., Zhang C., Kang N., Li Z. iVPF: Numerical Invertible Volume Preserving Flow for Efficient Lossless Compression, In 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA; 19–25 June 2021; 1–10. https://doi.org/10.1109/CVPR46437.2021.00068
27. Bai Y., Liu X., Wang K., Ji X., Wu X., Gao W. Deep lossy plus residual coding for lossless and near-lossless image compression. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2024; 46(5): 3577–3594. https://doi.org/10.48550/arXiv.2209.04847
28. Matsuda I., Ozaki N., Umezu Y., Itoh S. Lossless Coding Using Variable Blok-Size Adaptive Prediction Optimized for Each Image. Proceedings of 13th European Signal Processing Conference EUSIPCO 2005, 4–8 September 2005.
29. Ulacha G., Łazoryszczak M. Lossless image compression using context-dependent linear prediction based on absolute error minimization. Entropy. 2024; 26(120): 1–23. https://doi.org/10.3390/e26121115
30. Golchin F., Paliwal K. K. Classified adaptive prediction and entropy coding for lossless coding of images. In: Proceedings of International Conference on Image Processing, San-ta Barbara, USA, 26-29 October 1997; 3: 110–113. https://doi.org/10.1109/ICIP.1997.632006
31. Matsuda I., Mori H., Itoh S. Design of a minimum-rate predictor and its application to lossless image coding, 2000 10th European Signal Processing Conference, Tampere, Finland, 2000, 1–4.
32. Memon N.D., Sayood K. An Asymmetric lossless image compression technique. In: Proceedings of the 1995 International Conference on

Image Processing, Washington, DC, USA; 23-26 October 1995; 3: 97–100. https://doi.org/10.1109/ICIP.1995.537589

33. Aiazzi B., Baronti S., Alparone L. Near-lossless image compression by relaxation-labeled prediction, Signal Processing, November 2002; 82(11): 1619–1631. https://doi.org/10.1016/S0165-1684(02)00305-5
34. Frydrychowicz M, Ulacha G. Two-stage golomb – context-adaptive binary arithmetic coders coding in lossless image compression. Advances in Science and Technology Research Journal. 2024; 18(6): 62–85. https://doi.org/10.12913/22998624/191111
35. Hashidume Y., Morikawa Y. Lossless Image Coding Based on Minimum Mean Absolute Error Predictors. Proceedings of SICE Annual Conference 2007, Kagawa University, Japan, September 2007; 17–20: 2832–2836. https://doi.org/10.1109/SICE.2007.4421471
36. Dataset of 45 Images. 2022. (Accessed: 14.09.2025). https://kakit.zut.edu.pl/fileadmin/Test_Images.zip
37. Hsieh F.Y., Wang C.M., Lee C.C., Fan K.C. A lossless image coder integrating predictors and block-adaptive prediction. Journal of Information Science and Engineering. September 2008; 24(5): 1579–1591.
38. Dai W., Xiong H. Gaussian Process Regression Based Prediction for Lossless Image Coding. Proceedings of Data Compression Conference, Snowbird, Utah, USA, March 2014; 93–102. https://doi.org/10.1109/DCC.2014.72
39. Dai W., Xiong H., Wang J., Zheng Y.F., Large Discriminative Structured Set Prediction Modeling With Max-Margin Markov Network for Lossless Image Coding. IEEE Transactions on Image Processing. February 2014; 23(2): 541–554. https://doi.org/10.1109/tip.2013.2293429
40. VQ-MMAE$_2$ https://mega.nz/folder/u5A1SJRQ#E86RY-lntTTGIBfM-u3w-Q (Accessed: 11.11.2025)
41. Frydrychowicz M., Ulacha G. Lossless image compression method using vector quantization based on minimizing mean absolute error. Bulletin of the Polish Academy of Sciences Technical Sciences. June 2025; 73(5): e1554734–e1554734. https://doi.org/10.24425/bpasts.2025.154734
42. Strutz T. Multiplierless Reversible color transforms and their automatic selection for image data compression, IEEE Trans. Cir. and Sys. for Video Technol., July 2013; 23(7): 1249–1259. https://doi.org/10.1109/TCSVT.2013.2242612
43. Codec WebP 1.6, (Accessed: 14.09.2025). https://storage.googleapis.com/downloads.webmproject.org/releases/webp/libwebp-1.6.0-windows-x64.zip
44. Codec WebP2, (Accessed: 14.09.2025). https://chromium.googlesource.com/codecs/libwebp2/+/02b3132c3a9c40a9e544831b565eafca0e10a038
45. Codec Pngcrush 1.8.11, (Accessed: 14.09.2025). https://source-forge.net/projects/pmt/files/pngcrush-executables/1.8.11/
46. Codec JPEG-XL 0.9.1, (Accessed: 14.09.2025). https://github.com/libjxl/libjxl/releases/download/v0.9.1/jxl-x86-win dows-static.zip
47. AVIF 1.3.0 https://github.com/AOMediaCodec/libavif/releases/tag/v1.3.0 (Accessed: 9.11.2025)
48. FLIF 0.4 https://github.com/FLIF-hub/FLIF/releases/tag/v0.4 (Accessed: 9.11.2025)
49. Codec MRP 0.5, (Accessed: 14.09.2025). https://www.rs.tus.ac.jp/matsuda-lab/matsuda/mrp/mrp-05.tar.gz